Some group-based cryptosystems

Enric Ventura

Departament de Matemàtica Aplicada III

Universitat Politècnica de Catalunya

Zaragoza, January 23, 2009

Outline

Outline

The goal

$$\boxed{BOB} \quad \xrightarrow{m} \quad \boxed{ALICE}$$

Bob wants to send a secret message, m, to Alice over an open chanel (and Eve is trying to illegitimately discover m and break the system).

From Wikipedia: *"Diffie-Hellman key agreement was invented in 1976 ... and was the first practical method for establishing a shared secret over an unprotected communications chanel".*

A third author, Merkle, was also involved in the construction (U.S. Patent 4.200.770, now expired, describes the algorithms and credits Diffie, Hellman and Merkle as inventors).

The goal

$$\boxed{BOB} \quad \xrightarrow{m} \quad \boxed{ALICE}$$

Bob wants to send a secret message, m, to Alice over an open chanel (and Eve is trying to illegitimately discover m and break the system).

From Wikipedia: *"Diffie-Hellman key agreement was invented in 1976 ... and was the first practical method for establishing a shared secret over an unprotected communications chanel".*

A third author, Merkle, was also involved in the construction (U.S. Patent 4.200.770, now expired, describes the algorithms and credits Diffie, Hellman and Merkle as inventors).

The goal

$$\boxed{BOB} \quad \xrightarrow{m} \quad \boxed{ALICE}$$

Bob wants to send a secret message, m, to Alice over an open chanel (and Eve is trying to illegitimately discover m and break the system).

From Wikipedia: *"Diffie-Hellman key agreement was invented in 1976 ... and was the first practical method for establishing a shared secret over an unprotected communications chanel".*

A third author, Merkle, was also involved in the construction (U.S. Patent 4.200.770, now expired, describes the algorithms and credits Diffie, Hellman and Merkle as inventors).

Reduction to key establishment

- For simplicity, we assume that $m \in \{0,1\}^n$.
- Let $\mathcal{S}$ be a set and $H\colon \mathcal{S} \to \{0,1\}^n$ a function (called the key space and a Hash function, respectively).
- Suppose Bob and Alice share a secret key, $K \in \mathcal{S}$.
- **Encription:** Bob encrypts his message m as

$$E(m) = m + H(K).$$

- **Decryption:** Alice decrypts in the same way:

$$E(m) + H(K) = m + (H(K) + H(K)) = m.$$

- **Eavesdropper:** Eve needs to find $H(K)$, i.e. K.
- Expansion factor is 1.

Reduction to key establishment

- For simplicity, we assume that $m \in \{0,1\}^n$.
- Let $\mathcal{S}$ be a set and $H\colon \mathcal{S} \to \{0,1\}^n$ a function (called the key space and a Hash function, respectively).
- Suppose Bob and Alice share a secret key, $K \in \mathcal{S}$.
- **Encription:** Bob encrypts his message m as

$$E(m) = m + H(K).$$

- **Decryption:** Alice decrypts in the same way:

$$E(m) + H(K) = m + (H(K) + H(K)) = m.$$

- **Eavesdropper:** Eve needs to find $H(K)$, i.e. K.
- Expansion factor is 1.

Reduction to key establishment

- For simplicity, we assume that $m \in \{0,1\}^n$.
- Let $\mathcal{S}$ be a set and $H\colon \mathcal{S} \to \{0,1\}^n$ a function (called the key space and a Hash function, respectively).
- Suppose Bob and Alice share a secret key, $K \in \mathcal{S}$.
- **Encription:** Bob encrypts his message m as

$$E(m) = m + H(K).$$

- **Decryption:** Alice decrypts in the same way:

$$E(m) + H(K) = m + (H(K) + H(K)) = m.$$

- **Eavesdropper:** Eve needs to find $H(K)$, i.e. K.
- Expansion factor is 1.

Reduction to key establishment

- For simplicity, we assume that $m \in \{0,1\}^n$.
- Let $\mathcal{S}$ be a set and $H\colon \mathcal{S} \to \{0,1\}^n$ a function (called the key space and a Hash function, respectively).
- Suppose Bob and Alice share a secret key, $K \in \mathcal{S}$.
- **Encription:** Bob encrypts his message m as

$$E(m) = m + H(K).$$

- **Decryption:** Alice decrypts in the same way:

$$E(m) + H(K) = m + (H(K) + H(K)) = m.$$

- **Eavesdropper:** Eve needs to find $H(K)$, i.e. K.
- Expansion factor is 1.

Reduction to key establishment

- For simplicity, we assume that $m \in \{0,1\}^n$.
- Let $\mathcal{S}$ be a set and $H\colon \mathcal{S} \to \{0,1\}^n$ a function (called the key space and a Hash function, respectively).
- Suppose Bob and Alice share a secret key, $K \in \mathcal{S}$.
- **Encription:** Bob encrypts his message m as

$$E(m) = m + H(K).$$

- **Decryption:** Alice decrypts in the same way:

$$E(m) + H(K) = m + (H(K) + H(K)) = m.$$

- **Eavesdropper:** Eve needs to find $H(K)$, i.e. K.
- Expansion factor is 1.

Reduction to key establishment

- For simplicity, we assume that $m \in \{0,1\}^n$.
- Let $\mathcal{S}$ be a set and $H\colon \mathcal{S} \to \{0,1\}^n$ a function (called the key space and a Hash function, respectively).
- Suppose Bob and Alice share a secret key, $K \in \mathcal{S}$.
- **Encription:** Bob encrypts his message m as

$$E(m) = m + H(K).$$

- **Decryption:** Alice decrypts in the same way:

$$E(m) + H(K) = m + (H(K) + H(K)) = m.$$

- **Eavesdropper:** Eve needs to find $H(K)$, i.e. K.
- Expansion factor is 1.

Reduction to key establishment

- For simplicity, we assume that $m \in \{0,1\}^n$.
- Let $\mathcal{S}$ be a set and $H\colon \mathcal{S} \to \{0,1\}^n$ a function (called the key space and a Hash function, respectively).
- Suppose Bob and Alice share a secret key, $K \in \mathcal{S}$.
- **Encription:** Bob encrypts his message m as

$$E(m) = m + H(K).$$

- **Decryption:** Alice decrypts in the same way:

$$E(m) + H(K) = m + (H(K) + H(K)) = m.$$

- **Eavesdropper:** Eve needs to find $H(K)$, i.e. K.
- Expansion factor is 1.

Diffie-Hellman key exchange protocol (1976)

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Alice: picks a random $a \in \mathbb{N}$, and sends $g^a \mod p$.
- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$.
- Common secret: Alice: $(g^b)^a = g^{ba} \mod p$
  Bob: $(g^a)^b = g^{ab} \mod p$.
- Eve: knows $p,\ g$ and $g^a,\ g^b \mod p$, and needs $g^{ab} \mod p$.
- The protocol is considered to be secure against eavesdroppers, if p and g are chosen properly.

Diffie-Hellman key exchange protocol (1976)

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Alice: picks a random $a \in \mathbb{N}$, and sends $g^a \mod p$.
- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$.
- Common secret: Alice: $(g^b)^a = g^{ba} \mod p$
  Bob: $(g^a)^b = g^{ab} \mod p$.
- Eve: knows p, g and g^a, $g^b \mod p$, and needs $g^{ab} \mod p$.
- The protocol is considered to be secure against eavesdroppers, if p and g are chosen properly.

Diffie-Hellman key exchange protocol (1976)

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Alice: picks a random $a \in \mathbb{N}$, and sends $g^a \mod p$.
- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$.
- Common secret: Alice: $(g^b)^a = g^{ba} \mod p$
  Bob: $(g^a)^b = g^{ab} \mod p$.
- Eve: knows p, g and g^a, $g^b \mod p$, and needs $g^{ab} \mod p$.
- The protocol is considered to be secure against eavesdroppers, if p and g are chosen properly.

Diffie-Hellman key exchange protocol (1976)

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Alice: picks a random $a \in \mathbb{N}$, and sends $g^a \mod p$.
- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$.
- Common secret: Alice: $(g^b)^a = g^{ba} \mod p$
  Bob: $(g^a)^b = g^{ab} \mod p$.
- Eve: knows p, g and g^a, $g^b \mod p$, and needs $g^{ab} \mod p$.
- The protocol is considered to be secure against eavesdroppers, if p and g are chosen properly.

Diffie-Hellman key exchange protocol (1976)

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Alice: picks a random $a \in \mathbb{N}$, and sends $g^a \mod p$.
- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$.
- Common secret: Alice: $(g^b)^a = g^{ba} \mod p$
  Bob: $(g^a)^b = g^{ab} \mod p$.
- Eve: knows p, g and g^a, $g^b \mod p$, and needs $g^{ab} \mod p$.
- The protocol is considered to be secure against eavesdroppers, if p and g are chosen properly.

Diffie-Hellman key exchange protocol (1976)

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Alice: picks a random $a \in \mathbb{N}$, and sends $g^a \mod p$.
- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$.
- Common secret: Alice: $(g^b)^a = g^{ba} \mod p$
  Bob: $(g^a)^b = g^{ab} \mod p$.
- Eve: knows $p,\ g$ and $g^a,\ g^b \mod p$, and needs $g^{ab} \mod p$.
- The protocol is considered to be secure against eavesdroppers, if p and g are chosen properly.

Diffie-Hellman key exchange protocol (1976)

Eve needs to solve the

- Diffie-Hellman Problem: "knowing $p, \; g$ and $g^a, \; g^b \mod p$, compute $g^{ab} \mod p$",

or the

- Discrete Logarithm Problem: "knowing $p, \; g$ and $g^a \mod p$, compute a",

both currently considered to be "difficult" problems (but not known to be equivalent...).

Diffie-Hellman key exchange protocol (1976)

Eve needs to solve the

- Diffie-Hellman Problem: "knowing $p, \ g$ and $g^a, \ g^b \mod p$, compute $g^{ab} \mod p$",

or the

- Discrete Logarithm Problem: "knowing $p, \ g$ and $g^a \mod p$, compute a",

both currently considered to be "difficult" problems (but not known to be equivalent...).

Diffie-Hellman key exchange protocol (1976)

Eve needs to solve the

- Diffie-Hellman Problem: "knowing $p, \ g$ and $g^a, \ g^b \mod p$, compute $g^{ab} \mod p$",

or the

- Discrete Logarithm Problem: "knowing $p, \ g$ and $g^a \mod p$, compute a",

both currently considered to be "difficult" problems (but not known to be equivalent...).

Diffie-Hellman key exchange protocol (1976)

Brute force search for solving the Discrete Logarithm Problem requires computing $g, g^2, g^3, \ldots, g^{|g|} = 1$ (eventually, till $|g|$, the order of g modulo p): this is $O(|g|)$ multiplications.

In practical implementations, $|g|$ is typically about 10^{300}, so brute force attack is computationally infeasible.

This is not a problem for Alice and Bob because computing $g^a \mod p$ for a particular a is much faster, $O(\log_2 a)$, by the square-and-multiply method:

$$g^{21} = g^{16} \cdot g^4 \cdot g = (((g^2)^2)^2)^2 \cdot (g^2)^2 \cdot g.$$

Diffie-Hellman key exchange protocol (1976)

Brute force search for solving the Discrete Logarithm Problem requires computing $g, g^2, g^3, \ldots, g^{|g|} = 1$ (eventually, till $|g|$, the order of g modulo p): this is $O(|g|)$ multiplications.

In practical implementations, $|g|$ is typically about 10^{300}, so brute force attack is computationally infeasible.

This is not a problem for Alice and Bob because computing $g^a \mod p$ for a particular a is much faster, $O(\log_2 a)$, by the square-and-multiply method:

$$g^{21} = g^{16} \cdot g^4 \cdot g = (((g^2)^2)^2)^2 \cdot (g^2)^2 \cdot g.$$

Diffie-Hellman key exchange protocol (1976)

Brute force search for solving the Discrete Logarithm Problem requires computing $g, g^2, g^3, \ldots, g^{|g|} = 1$ (eventually, till $|g|$, the order of g modulo p): this is $O(|g|)$ multiplications.

In practical implementations, $|g|$ is typically about 10^{300}, so brute force attack is computationally infeasible.

This is not a problem for Alice and Bob because computing $g^a \mod p$ for a particular a is much faster, $O(\log_2 a)$, by the square-and-multiply method:

$$g^{21} = g^{16} \cdot g^4 \cdot g = (((g^2)^2)^2)^2 \cdot (g^2)^2 \cdot g.$$

Diffie-Hellman key exchange protocol (1976)

Brute force search for solving the Discrete Logarithm Problem requires computing $g, g^2, g^3, \ldots, g^{|g|} = 1$ (eventually, till $|g|$, the order of g modulo p): this is $O(|g|)$ multiplications.

In practical implementations, $|g|$ is typically about 10^{300}, so brute force attack is computationally infeasible.

This is not a problem for Alice and Bob because computing $g^a \mod p$ for a particular a is much faster, $O(\log_2 a)$, by the square-and-multiply method:

$$g^{21} = g^{16} \cdot g^4 \cdot g = (((g^2)^2)^2)^2 \cdot (g^2)^2 \cdot g.$$

Outline

The word problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Word Problem: "given a word $w(x_1, \ldots, x_n)$ decide whether $w =_G 1$ or not (i.e. whether $w \in \ll R \gg$)".

There are finitely presented groups with unsolvable Word Problem.

A set of words Σ on X is said to have no collision in G if the natural map $\Sigma \to G$ is injective.

The word problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Word Problem: "given a word $w(x_1, \ldots, x_n)$ decide whether $w =_G 1$ or not (i.e. whether $w \in \ll R \gg$)".

There are finitely presented groups with unsolvable Word Problem.

A set of words Σ on X is said to have no collision in G if the natural map $\Sigma \to G$ is injective.

The word problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Word Problem: "given a word $w(x_1, \ldots, x_n)$ decide whether $w =_G 1$ or not (i.e. whether $w \in \ll R \gg$)".

There are finitely presented groups with unsolvable Word Problem.

A set of words Σ on X is said to have no collision in G if the natural map $\Sigma \to G$ is injective.

The word problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Word Problem: "given a word $w(x_1, \ldots, x_n)$ decide whether $w =_G 1$ or not (i.e. whether $w \in \ll R \gg$)".

There are finitely presented groups with unsolvable Word Problem.

A set of words Σ on X is said to have no collision in G if the natural map $\Sigma \to G$ is injective.

Wagner-Magyarik protocol (1984)

- Public: A platform $G = \langle X \mid R \rangle$ and two words $\Sigma = \{w_0, w_1\}$.
- Private: A set of words S such that
  - the Word Problem is "difficult" in $G = \langle X \mid R \rangle$,
  - the Word Problem is "easy" in $G' = \langle X, R \cup S \rangle = G/S$,
  - Σ has no collision in G' (and so, in G).
- Bob: encodes each bit b in his message by an arbitrary (and changing) word w such that $w =_G w_b$.
- Alice: decodes w by solving the Word Problem in G': decide whether $w =_{G'} w_0$ or $w =_{G'} w_1$.
- Eve: sees w and needs to decide whether $w =_G w_0$ or $w =_G w_1$. This is the Word Problem in G.

Wagner-Magyarik protocol (1984)

- Public: A platform $G = \langle X \mid R \rangle$ and two words $\Sigma = \{w_0, w_1\}$.
- Private: A set of words S such that
  - the Word Problem is "difficult" in $G = \langle X \mid R \rangle$,
  - the Word Problem is "easy" in $G' = \langle X, R \cup S \rangle = G/S$,
  - Σ has no collision in G' (and so, in G).
- Bob: encodes each bit b in his message by an arbitrary (and changing) word w such that $w =_G w_b$.
- Alice: decodes w by solving the Word Problem in G': decide whether $w =_{G'} w_0$ or $w =_{G'} w_1$.
- Eve: sees w and needs to decide whether $w =_G w_0$ or $w =_G w_1$. This is the Word Problem in G.

Wagner-Magyarik protocol (1984)

- Public: A platform $G = \langle X \mid R \rangle$ and two words $\Sigma = \{w_0, w_1\}$.
- Private: A set of words S such that
  - the Word Problem is "difficult" in $G = \langle X \mid R \rangle$,
  - the Word Problem is "easy" in $G' = \langle X, R \cup S \rangle = G/S$,
  - Σ has no collision in G' (and so, in G).
- Bob: encodes each bit b in his message by an arbitrary (and changing) word w such that $w =_G w_b$.
- Alice: decodes w by solving the Word Problem in G': decide whether $w =_{G'} w_0$ or $w =_{G'} w_1$.
- Eve: sees w and needs to decide whether $w =_G w_0$ or $w =_G w_1$. This is the Word Problem in G.

Wagner-Magyarik protocol (1984)

- Public: A platform $G = \langle X \mid R \rangle$ and two words $\Sigma = \{w_0, w_1\}$.
- Private: A set of words S such that
  - the Word Problem is "difficult" in $G = \langle X \mid R \rangle$,
  - the Word Problem is "easy" in $G' = \langle X, R \cup S \rangle = G/S$,
  - Σ has no collision in G' (and so, in G).
- Bob: encodes each bit b in his message by an arbitrary (and changing) word w such that $w =_G w_b$.
- Alice: decodes w by solving the Word Problem in G': decide whether $w =_{G'} w_0$ or $w =_{G'} w_1$.
- Eve: sees w and needs to decide whether $w =_G w_0$ or $w =_G w_1$. This is the Word Problem in G.

Wagner-Magyarik protocol (1984)

- Public: A platform $G = \langle X \mid R \rangle$ and two words $\Sigma = \{w_0, w_1\}$.
- Private: A set of words S such that
  - the Word Problem is "difficult" in $G = \langle X \mid R \rangle$,
  - the Word Problem is "easy" in $G' = \langle X, R \cup S \rangle = G/S$,
  - Σ has no collision in G' (and so, in G).
- Bob: encodes each bit b in his message by an arbitrary (and changing) word w such that $w =_G w_b$.
- Alice: decodes w by solving the Word Problem in G': decide whether $w =_{G'} w_0$ or $w =_{G'} w_1$.
- Eve: sees w and needs to decide whether $w =_G w_0$ or $w =_G w_1$. This is the Word Problem in G.

Wagner-Magyarik protocol (1984)

- Public: A platform $G = \langle X \mid R \rangle$ and two words $\Sigma = \{w_0, w_1\}$.
- Private: A set of words S such that
  - the Word Problem is "difficult" in $G = \langle X \mid R \rangle$,
  - the Word Problem is "easy" in $G' = \langle X, R \cup S \rangle = G/S$,
  - Σ has no collision in G' (and so, in G).
- Bob: encodes each bit b in his message by an arbitrary (and changing) word w such that $w =_G w_b$.
- Alice: decodes w by solving the Word Problem in G': decide whether $w =_{G'} w_0$ or $w =_{G'} w_1$.
- Eve: sees w and needs to decide whether $w =_G w_0$ or $w =_G w_1$. This is the Word Problem in G.

Wagner-Magyarik protocol (1984)

- Public: A platform $G = \langle X \mid R \rangle$ and two words $\Sigma = \{w_0, w_1\}$.
- Private: A set of words S such that
  - the Word Problem is "difficult" in $G = \langle X \mid R \rangle$,
  - the Word Problem is "easy" in $G' = \langle X, R \cup S \rangle = G/S$,
  - Σ has no collision in G' (and so, in G).
- Bob: encodes each bit b in his message by an arbitrary (and changing) word w such that $w =_G w_b$.
- Alice: decodes w by solving the Word Problem in G': decide whether $w =_{G'} w_0$ or $w =_{G'} w_1$.
- Eve: sees w and needs to decide whether $w =_G w_0$ or $w =_G w_1$. This is the Word Problem in G.

Wagner-Magyarik protocol (1984)

- Public: A platform $G = \langle X \mid R \rangle$ and two words $\Sigma = \{w_0, w_1\}$.
- Private: A set of words S such that
  - the Word Problem is "difficult" in $G = \langle X \mid R \rangle$,
  - the Word Problem is "easy" in $G' = \langle X, R \cup S \rangle = G/S$,
  - Σ has no collision in G' (and so, in G).
- Bob: encodes each bit b in his message by an arbitrary (and changing) word w such that $w =_G w_b$.
- Alice: decodes w by solving the Word Problem in G': decide whether $w =_{G'} w_0$ or $w =_{G'} w_1$.
- Eve: sees w and needs to decide whether $w =_G w_0$ or $w =_G w_1$. This is the Word CHOICE Problem in G.
- Or...: find an alternative private key, T, with easy Word Problem in G/T, and no collision for Σ.

Wagner-Magyarik protocol (1984)

- Public: A platform $G = \langle X \mid R \rangle$ and two words $\Sigma = \{w_0, w_1\}$.
- Private: A set of words S such that
  - the Word Problem is "difficult" in $G = \langle X \mid R \rangle$,
  - the Word Problem is "easy" in $G' = \langle X, R \cup S \rangle = G/S$,
  - Σ has no collision in G' (and so, in G).
- Bob: encodes each bit b in his message by an arbitrary (and changing) word w such that $w =_G w_b$.
- Alice: decodes w by solving the Word Problem in G': decide whether $w =_{G'} w_0$ or $w =_{G'} w_1$.
- Eve: sees w and needs to decide whether $w =_G w_0$ or $w =_G w_1$. This is the Word CHOICE Problem in G.
- Or...: find an alternative private key, T, with easy Word Problem in G/T, and no collision for Σ.

Outline

The conjugacy problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Conjugacy Problem: "given $u, v \in G$ (as words on the x_i's), decide whether $v =_G x^{-1}ux$ for some $x \in G$".

Solvable Conjugacy Problem $\Longrightarrow$ solvable Word Problem.

Solvable Conjugacy Problem $\not\Longleftarrow$ solvable Word Problem.

- Conjugacy Search Problem: "given $u, v \in G$ and the information that u and v are conjugate to each other in G, find an $x \in G$ such that $v =_G x^{-1}ux$".

CSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The conjugacy problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Conjugacy Problem: "given $u, v \in G$ (as words on the x_i's), decide whether $v =_G x^{-1}ux$ for some $x \in G$".

Solvable Conjugacy Problem $\Longrightarrow$ solvable Word Problem.

Solvable Conjugacy Problem $\not\Longleftarrow$ solvable Word Problem.

- Conjugacy Search Problem: "given $u, v \in G$ and the information that u and v are conjugate to each other in G, find an $x \in G$ such that $v =_G x^{-1}ux$".

CSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The conjugacy problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Conjugacy Problem: "given $u, v \in G$ (as words on the x_i's), decide whether $v =_G x^{-1}ux$ for some $x \in G$".

Solvable Conjugacy Problem $\Longrightarrow$ solvable Word Problem.

Solvable Conjugacy Problem $\not\Longleftarrow$ solvable Word Problem.

- Conjugacy Search Problem: "given $u, v \in G$ and the information that u and v are conjugate to each other in G, find an $x \in G$ such that $v =_G x^{-1}ux$".

CSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The conjugacy problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Conjugacy Problem: "given $u, v \in G$ (as words on the x_i's), decide whether $v =_G x^{-1}ux$ for some $x \in G$".

Solvable Conjugacy Problem $\Longrightarrow$ solvable Word Problem.

Solvable Conjugacy Problem $\not\Longleftarrow$ solvable Word Problem.

- Conjugacy Search Problem: "given $u, v \in G$ and the information that u and v are conjugate to each other in G, find an $x \in G$ such that $v =_G x^{-1}ux$".

CSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The conjugacy problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Conjugacy Problem: "given $u, v \in G$ (as words on the x_i's), decide whether $v =_G x^{-1}ux$ for some $x \in G$".

Solvable Conjugacy Problem $\Longrightarrow$ solvable Word Problem.

Solvable Conjugacy Problem $\not\Longleftarrow$ solvable Word Problem.

- Conjugacy Search Problem: "given $u, v \in G$ and the information that u and v are conjugate to each other in G, find an $x \in G$ such that $v =_G x^{-1}ux$".

CSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The conjugacy problem in groups

Let $\langle x_1, \ldots, x_n \mid r_1, \ldots, r_m \rangle$ be a finite presentation of a group G.

- Conjugacy Problem: "given $u, v \in G$ (as words on the x_i's), decide whether $v =_G x^{-1}ux$ for some $x \in G$".

Solvable Conjugacy Problem $\Longrightarrow$ solvable Word Problem.

Solvable Conjugacy Problem $\not\Longleftarrow$ solvable Word Problem.

- Conjugacy Search Problem: "given $u, v \in G$ and the information that u and v are conjugate to each other in G, find an $x \in G$ such that $v =_G x^{-1}ux$".

CSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The conjugacy problem in groups

- Multiple Conjugacy Problem: given $u_1, \ldots u_k,\ v_1, \ldots v_k \in G$, decide whether $\exists x \in G$ such that $v_i =_G x^{-1} u_i x,\ \forall i$.

Solv. Multiple Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Multiple Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Restricted Conjugacy Problem: "given u, v and a subgroup $H \leqslant G$, decide whether $v =_G x^{-1} ux$ for some $x \in H$".

We can consider all variations search/non-search, multiple/simple, restricted/non-restricted.

The conjugacy problem in groups

- Multiple Conjugacy Problem: given $u_1, \ldots u_k,\ v_1, \ldots v_k \in G$, decide whether $\exists x \in G$ such that $v_i =_G x^{-1} u_i x,\ \forall i$.

Solv. Multiple Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Multiple Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Restricted Conjugacy Problem: "given u, v and a subgroup $H \leqslant G$, decide whether $v =_G x^{-1} u x$ for some $x \in H$".

We can consider all variations search/non-search, multiple/simple, restricted/non-restricted.

The conjugacy problem in groups

- Multiple Conjugacy Problem: given $u_1, \ldots u_k,\ v_1, \ldots v_k \in G$, decide whether $\exists x \in G$ such that $v_i =_G x^{-1} u_i x,\ \forall i$.

Solv. Multiple Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Multiple Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Restricted Conjugacy Problem: "given u, v and a subgroup $H \leqslant G$, decide whether $v =_G x^{-1} ux$ for some $x \in H$".

We can consider all variations search/non-search, multiple/simple, restricted/non-restricted.

The conjugacy problem in groups

- Multiple Conjugacy Problem: given $u_1, \ldots u_k,\ v_1, \ldots v_k \in G$, decide whether $\exists x \in G$ such that $v_i =_G x^{-1} u_i x,\ \forall i$.

Solv. Multiple Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Multiple Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Restricted Conjugacy Problem: "given u, v and a subgroup $H \leqslant G$, decide whether $v =_G x^{-1} u x$ for some $x \in H$".

We can consider all variations search/non-search, multiple/simple, restricted/non-restricted.

The conjugacy problem in groups

- Multiple Conjugacy Problem: given $u_1, \ldots u_k,\ v_1, \ldots v_k \in G$, decide whether $\exists x \in G$ such that $v_i =_G x^{-1} u_i x,\ \forall i$.

Solv. Multiple Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Multiple Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Restricted Conjugacy Problem: “given u, v and a subgroup $H \leqslant G$, decide whether $v =_G x^{-1} ux$ for some $x \in H$”.

We can consider all variations search/non-search, multiple/simple, restricted/non-restricted.

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Public: $G = \langle X \mid R \rangle$, $w \in G$, and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A, \forall b \in B$.
- Alice: picks a random $a \in A$, and sends $a^{-1}wa = w^a$.
- Bob: picks a random $b \in B$, and sends $b^{-1}wb = w^b$.
- Common secret: Alice: $a^{-1}(b^{-1}wb)a = w^{ba}$, Bob: $b^{-1}(a^{-1}wa)b = w^{ab}$.
- Eve: knows w, w^a, w^b, and needs w^{ab}.
  This can be done by solving the Conjugacy Search Problem Restricted to A (or B),

  ... but also solving the following seemingly easier problem:

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Public: $G = \langle X \mid R \rangle$, $w \in G$, and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A, \forall b \in B$.
- Alice: picks a random $a \in A$, and sends $a^{-1}wa = w^a$.
- Bob: picks a random $b \in B$, and sends $b^{-1}wb = w^b$.
- Common secret: Alice: $a^{-1}(b^{-1}wb)a = w^{ba}$, Bob: $b^{-1}(a^{-1}wa)b = w^{ab}$.
- Eve: knows w, w^a, w^b, and needs w^{ab}.
This can be done by solving the Conjugacy Search Problem Restricted to A (or B),

... but also solving the following seemingly easier problem:

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Public: $G = \langle X \mid R \rangle$, $w \in G$, and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Alice: picks a random $a \in A$, and sends $a^{-1}wa = w^a$.
- Bob: picks a random $b \in B$, and sends $b^{-1}wb = w^b$.
- Common secret: Alice: $a^{-1}(b^{-1}wb)a = w^{ba}$,
  Bob: $b^{-1}(a^{-1}wa)b = w^{ab}$.
- Eve: knows w, w^a, w^b, and needs w^{ab}.
  This can be done by solving the Conjugacy Search Problem Restricted to A (or B),

  ... but also solving the following seemingly easier problem:

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Public: $G = \langle X \mid R \rangle$, $w \in G$, and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A, \forall b \in B$.
- Alice: picks a random $a \in A$, and sends $a^{-1}wa = w^a$.
- Bob: picks a random $b \in B$, and sends $b^{-1}wb = w^b$.
- Common secret: Alice: $a^{-1}(b^{-1}wb)a = w^{ba}$,
  Bob: $b^{-1}(a^{-1}wa)b = w^{ab}$.
- Eve: knows w, w^a, w^b, and needs w^{ab}.
  This can be done by solving the Conjugacy Search Problem Restricted to A (or B),

  ... but also solving the following seemingly easier problem:

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Public: $G = \langle X \mid R \rangle$, $w \in G$, and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A, \forall b \in B$.
- Alice: picks a random $a \in A$, and sends $a^{-1}wa = w^a$.
- Bob: picks a random $b \in B$, and sends $b^{-1}wb = w^b$.
- Common secret: Alice: $a^{-1}(b^{-1}wb)a = w^{ba}$,
  Bob: $b^{-1}(a^{-1}wa)b = w^{ab}$.
- Eve: knows w, w^a, w^b, and needs w^{ab}.
  This can be done by solving the Conjugacy Search Problem Restricted to A (or B),

  ... but also solving the following seemingly easier problem:

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Public: $G = \langle X \mid R \rangle$, $w \in G$, and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A, \forall b \in B$.
- Alice: picks a random $a \in A$, and sends $a^{-1}wa = w^a$.
- Bob: picks a random $b \in B$, and sends $b^{-1}wb = w^b$.
- Common secret: Alice: $a^{-1}(b^{-1}wb)a = w^{ba}$,
  Bob: $b^{-1}(a^{-1}wa)b = w^{ab}$.
- Eve: knows w, w^a, w^b, and needs w^{ab}.
  This can be done by solving the Conjugacy Search Problem Restricted to A (or B),

... but also solving the following seemingly easier problem:

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Public: $G = \langle X \mid R \rangle$, $w \in G$, and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A, \forall b \in B$.
- Alice: picks a random $a \in A$, and sends $a^{-1}wa = w^a$.
- Bob: picks a random $b \in B$, and sends $b^{-1}wb = w^b$.
- Common secret: Alice: $a^{-1}(b^{-1}wb)a = w^{ba}$,
  Bob: $b^{-1}(a^{-1}wa)b = w^{ab}$.
- Eve: knows w, w^a, w^b, and needs w^{ab}.
  This can be done by solving the Conjugacy Search Problem Restricted to A (or B),

  ... but also solving the following seemingly easier problem:

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Decomposition Problem: "knowing $w, w' \in G$, find $a_1, a_2 \in A$ such that $w' = a_1 w a_2$".

Eve knows w, w^a, w^b and suppose she can compute $a_1, a_2 \in A$ such that $w^a = a_1 w a_2$.

Then, $a_1 w^b a_2 = a_1 (b^{-1} w b) a_2 = b^{-1} (a_1 w a_2) b = b^{-1} w^a b = w^{ab}$, and she finds the secret.

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Decomposition Problem: "knowing $w, w' \in G$, find $a_1, a_2 \in A$ such that $w' = a_1 w a_2$".

Eve knows w, w^a, w^b and suppose she can compute $a_1, a_2 \in A$ such that $w^a = a_1 w a_2$.

Then, $a_1 w^b a_2 = a_1(b^{-1}wb)a_2 = b^{-1}(a_1 w a_2)b = b^{-1}w^a b = w^{ab}$, and she finds the secret.

Ko-Lee-Cheon-Han-Kang-Park Protocol (2000)

- Decomposition Problem: "knowing $w, w' \in G$, find $a_1, a_2 \in A$ such that $w' = a_1 w a_2$".

Eve knows w, w^a, w^b and suppose she can compute $a_1, a_2 \in A$ such that $w^a = a_1 w a_2$.

Then, $a_1 w^b a_2 = a_1(b^{-1}wb)a_2 = b^{-1}(a_1 w a_2)b = b^{-1}w^a b = w^{ab}$, and she finds the secret.

Hiding one of the subgroups, Shpilrain-Ushakov (2006)

Shpilrain-Ushakov did the following variation of Ko-Lee protocol:

- Public: $G = \langle X \mid R \rangle$ and $w \in G$.
- Alice: picks a random $a_1 \in G$, a f.g. subgroup $A \leqslant C_G(a_1)$ and sends generators $A = \langle \alpha_1, \ldots, \alpha_n \rangle$.
- Bob: picks a random $b_2 \in B$, a f.g. subgroup $B \leqslant C_G(b_2)$ and sends generators $B = \langle \beta_1, \ldots, \beta_m \rangle$.
- Alice: picks a random $a_2 \in B$, and sends $a_1 w a_2$.
- Bob: picks a random $b_1 \in A$, and sends $b_1 w b_2$.
- Common secret: Alice: $a_1(b_1 w b_2)a_2$,
  Bob: $b_1(a_1 w a_2)b_2$.
- Eve: knows w, $a_1 w a_2$, $b_1 w b_2$, and needs $a_1 b_1 w a_2 b_2$.
  This can be done by trying to recover a_1 and a_2 from w and $a_1 w a_2$, and knowing that $a_2 \in B$, but without any information where to look for a_1.

Hiding one of the subgroups, Shpilrain-Ushakov (2006)

Shpilrain-Ushakov did the following variation of Ko-Lee protocol:

- Public: $G = \langle X \mid R \rangle$ and $w \in G$.
- Alice: picks a random $a_1 \in G$, a f.g. subgroup $A \leqslant C_G(a_1)$ and sends generators $A = \langle \alpha_1, \ldots, \alpha_n \rangle$.
- Bob: picks a random $b_2 \in B$, a f.g. subgroup $B \leqslant C_G(b_2)$ and sends generators $B = \langle \beta_1, \ldots, \beta_m \rangle$.
- Alice: picks a random $a_2 \in B$, and sends $a_1 w a_2$.
- Bob: picks a random $b_1 \in A$, and sends $b_1 w b_2$.
- Common secret: Alice: $a_1(b_1 w b_2)a_2$, Bob: $b_1(a_1 w a_2)b_2$.
- Eve: knows w, $a_1 w a_2$, $b_1 w b_2$, and needs $a_1 b_1 w a_2 b_2$. This can be done by trying to recover a_1 and a_2 from w and $a_1 w a_2$, and knowing that $a_2 \in B$, but without any information where to look for a_1.

Hiding one of the subgroups, Shpilrain-Ushakov (2006)

Shpilrain-Ushakov did the following variation of Ko-Lee protocol:

- Public: $G = \langle X \mid R \rangle$ and $w \in G$.
- Alice: picks a random $a_1 \in G$, a f.g. subgroup $A \leqslant C_G(a_1)$ and sends generators $A = \langle \alpha_1, \ldots, \alpha_n \rangle$.
- Bob: picks a random $b_2 \in B$, a f.g. subgroup $B \leqslant C_G(b_2)$ and sends generators $B = \langle \beta_1, \ldots, \beta_m \rangle$.
- Alice: picks a random $a_2 \in B$, and sends $a_1 w a_2$.
- Bob: picks a random $b_1 \in A$, and sends $b_1 w b_2$.
- Common secret: Alice: $a_1(b_1 w b_2)a_2$,
  Bob: $b_1(a_1 w a_2)b_2$.
- Eve: knows w, $a_1 w a_2$, $b_1 w b_2$, and needs $a_1 b_1 w a_2 b_2$.
  This can be done by trying to recover a_1 and a_2 from w and $a_1 w a_2$, and knowing that $a_2 \in B$, but without any information where to look for a_1.

Hiding one of the subgroups, Shpilrain-Ushakov (2006)

Shpilrain-Ushakov did the following variation of Ko-Lee protocol:

- Public: $G = \langle X \mid R \rangle$ and $w \in G$.
- Alice: picks a random $a_1 \in G$, a f.g. subgroup $A \leqslant C_G(a_1)$ and sends generators $A = \langle \alpha_1, \ldots, \alpha_n \rangle$.
- Bob: picks a random $b_2 \in B$, a f.g. subgroup $B \leqslant C_G(b_2)$ and sends generators $B = \langle \beta_1, \ldots, \beta_m \rangle$.
- Alice: picks a random $a_2 \in B$, and sends $a_1 w a_2$.
- Bob: picks a random $b_1 \in A$, and sends $b_1 w b_2$.
- Common secret: Alice: $a_1(b_1 w b_2)a_2$,
  Bob: $b_1(a_1 w a_2)b_2$.
- Eve: knows w, $a_1 w a_2$, $b_1 w b_2$, and needs $a_1 b_1 w a_2 b_2$.
  This can be done by trying to recover a_1 and a_2 from w and $a_1 w a_2$, and knowing that $a_2 \in B$, but without any information where to look for a_1.

Hiding one of the subgroups, Shpilrain-Ushakov (2006)

Shpilrain-Ushakov did the following variation of Ko-Lee protocol:

- Public: $G = \langle X \mid R \rangle$ and $w \in G$.
- Alice: picks a random $a_1 \in G$, a f.g. subgroup $A \leqslant C_G(a_1)$ and sends generators $A = \langle \alpha_1, \ldots, \alpha_n \rangle$.
- Bob: picks a random $b_2 \in B$, a f.g. subgroup $B \leqslant C_G(b_2)$ and sends generators $B = \langle \beta_1, \ldots, \beta_m \rangle$.
- Alice: picks a random $a_2 \in B$, and sends $a_1 w a_2$.
- Bob: picks a random $b_1 \in A$, and sends $b_1 w b_2$.
- Common secret: Alice: $a_1(b_1 w b_2)a_2$,
  Bob: $b_1(a_1 w a_2)b_2$.
- Eve: knows w, $a_1 w a_2$, $b_1 w b_2$, and needs $a_1 b_1 w a_2 b_2$.
  This can be done by trying to recover a_1 and a_2 from w and $a_1 w a_2$, and knowing that $a_2 \in B$, but without any information where to look for a_1.

Hiding one of the subgroups, Shpilrain-Ushakov (2006)

Shpilrain-Ushakov did the following variation of Ko-Lee protocol:

- Public: $G = \langle X \mid R \rangle$ and $w \in G$.
- Alice: picks a random $a_1 \in G$, a f.g. subgroup $A \leqslant C_G(a_1)$ and sends generators $A = \langle \alpha_1, \ldots, \alpha_n \rangle$.
- Bob: picks a random $b_2 \in B$, a f.g. subgroup $B \leqslant C_G(b_2)$ and sends generators $B = \langle \beta_1, \ldots, \beta_m \rangle$.
- Alice: picks a random $a_2 \in B$, and sends $a_1 w a_2$.
- Bob: picks a random $b_1 \in A$, and sends $b_1 w b_2$.
- Common secret: Alice: $a_1(b_1 w b_2)a_2$,
  Bob: $b_1(a_1 w a_2)b_2$.
- Eve: knows w, $a_1 w a_2$, $b_1 w b_2$, and needs $a_1 b_1 w a_2 b_2$.
  This can be done by trying to recover a_1 and a_2 from w and $a_1 w a_2$, and knowing that $a_2 \in B$, but without any information where to look for a_1.

Hiding one of the subgroups, Shpilrain-Ushakov (2006)

Shpilrain-Ushakov did the following variation of Ko-Lee protocol:

- Public: $G = \langle X \mid R \rangle$ and $w \in G$.
- Alice: picks a random $a_1 \in G$, a f.g. subgroup $A \leqslant C_G(a_1)$ and sends generators $A = \langle \alpha_1, \ldots, \alpha_n \rangle$.
- Bob: picks a random $b_2 \in B$, a f.g. subgroup $B \leqslant C_G(b_2)$ and sends generators $B = \langle \beta_1, \ldots, \beta_m \rangle$.
- Alice: picks a random $a_2 \in B$, and sends $a_1 w a_2$.
- Bob: picks a random $b_1 \in A$, and sends $b_1 w b_2$.
- Common secret: Alice: $a_1(b_1 w b_2)a_2$,
  Bob: $b_1(a_1 w a_2)b_2$.
- Eve: knows w, $a_1 w a_2$, $b_1 w b_2$, and needs $a_1 b_1 w a_2 b_2$.
  This can be done by trying to recover a_1 and a_2 from w and $a_1 w a_2$, and knowing that $a_2 \in B$, but without any information where to look for a_1.

Hiding one of the subgroups, Shpilrain-Ushakov (2006)

Shpilrain-Ushakov did the following variation of Ko-Lee protocol:

- Public: $G = \langle X \mid R \rangle$ and $w \in G$.
- Alice: picks a random $a_1 \in G$, a f.g. subgroup $A \leqslant C_G(a_1)$ and sends generators $A = \langle \alpha_1, \ldots, \alpha_n \rangle$.
- Bob: picks a random $b_2 \in B$, a f.g. subgroup $B \leqslant C_G(b_2)$ and sends generators $B = \langle \beta_1, \ldots, \beta_m \rangle$.
- Alice: picks a random $a_2 \in B$, and sends $a_1 w a_2$.
- Bob: picks a random $b_1 \in A$, and sends $b_1 w b_2$.
- Common secret: Alice: $a_1 (b_1 w b_2) a_2$,
  Bob: $b_1 (a_1 w a_2) b_2$.
- Eve: knows w, $a_1 w a_2$, $b_1 w b_2$, and needs $a_1 b_1 w a_2 b_2$.
  This can be done by trying to recover a_1 and a_2 from w and $a_1 w a_2$, and knowing that $a_2 \in B$, but without any information where to look for a_1.

Hiding one of the subgroups, Shpilrain-Ushakov (2006)

Shpilrain-Ushakov did the following variation of Ko-Lee protocol:

- Public: $G = \langle X \mid R \rangle$ and $w \in G$.
- Alice: picks a random $a_1 \in G$, a f.g. subgroup $A \leqslant C_G(a_1)$ and sends generators $A = \langle \alpha_1, \ldots, \alpha_n \rangle$.
- Bob: picks a random $b_2 \in B$, a f.g. subgroup $B \leqslant C_G(b_2)$ and sends generators $B = \langle \beta_1, \ldots, \beta_m \rangle$.
- Alice: picks a random $a_2 \in B$, and sends $a_1 w a_2$.
- Bob: picks a random $b_1 \in A$, and sends $b_1 w b_2$.
- Common secret: Alice: $a_1(b_1 w b_2) a_2$,
  Bob: $b_1(a_1 w a_2) b_2$.
- Eve: knows w, $a_1 w a_2$, $b_1 w b_2$, and needs $a_1 b_1 w a_2 b_2$.
  This can be done by trying to recover a_1 and a_2 from w and $a_1 w a_2$, and knowing that $a_2 \in B$, but without any information where to look for a_1.

Outline

The factorization problem

- Factorization Problem: "given $u \in G$ and $A, B \leqslant G$, decide whether $u =_G ab$ for some $a \in A$ and $b \in B$".

- Factorization Search Problem: "given $u \in G$, $A, B \leqslant G$, and the information that $u = ab$ for some $a \in A$ and $b \in B$, find such a and b."

- Triple Factorization Search Problem: "given $u \in G$, $A, B, C \leqslant G$, and the information that $u = abc$ for some $a \in A$, $b \in B$ and $c \in C$, find such a, b and c."

The factorization problem

- Factorization Problem: "given $u \in G$ and $A, B \leqslant G$, decide whether $u =_G ab$ for some $a \in A$ and $b \in B$".

- Factorization Search Problem: "given $u \in G$, $A, B \leqslant G$, and the information that $u = ab$ for some $a \in A$ and $b \in B$, find such a and b."

- Triple Factorization Search Problem: "given $u \in G$, $A, B, C \leqslant G$, and the information that $u = abc$ for some $a \in A$, $b \in B$ and $c \in C$, find such a, b and c."

The factorization problem

- Factorization Problem: "given $u \in G$ and $A, B \leqslant G$, decide whether $u =_G ab$ for some $a \in A$ and $b \in B$".
- Factorization Search Problem: "given $u \in G$, $A, B \leqslant G$, and the information that $u = ab$ for some $a \in A$ and $b \in B$, find such a and b."
- Triple Factorization Search Problem: "given $u \in G$, $A, B, C \leqslant G$, and the information that $u = abc$ for some $a \in A$, $b \in B$ and $c \in C$, find such a, b and c."

The factorization problem

- Factorization Problem: "given $u \in G$ and $A, B \leqslant G$, decide whether $u =_G ab$ for some $a \in A$ and $b \in B$".
- Factorization Search Problem: "given $u \in G$, $A, B \leqslant G$, and the information that $u = ab$ for some $a \in A$ and $b \in B$, find such a and b."
- Triple Factorization Search Problem: "given $u \in G$, $A, B, C \leqslant G$, and the information that $u = abc$ for some $a \in A$, $b \in B$ and $c \in C$, find such a, b and c."

A protocol based on the Factorization Search Problem

- Public: $G = \langle X \mid R \rangle$ and $A, B \leqslant G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Alice: picks a random $a_1 \in A$, $b_1 \in B$ and sends $a_1 b_1$.
- Bob: picks a random $a_2 \in A$, $b_2 \in B$ and sends $a_2 b_2$.
- Common secret: Alice: $b_1(a_2 b_2)a_1 = a_2 b_1 b_2 a_1 = a_2 a_1 b_1 b_2$.
  Bob: $a_2(a_1 b_1)b_2$.
- Eve: knows $a_1 a_2$ and $b_1 b_2$, and needs $a_2 a_1 b_1 b_2$.
  This can be done by solving the Factorization Search Problem in A (or B).

  Note that Eve can compute

  $$(a_1 b_1)(a_2 b_2) = a_1 a_2 b_1 b_2 \quad \text{and} \quad (a_2 b_2)(a_1 b_1) = a_2 a_1 b_2 b_1,$$

  but neither of these products equal the secret if $a_1 a_2 \neq a_2 a_1$ and $b_1 b_2 \neq b_2 b_1$.

A protocol based on the Factorization Search Problem

- Public: $G = \langle X \mid R \rangle$ and $A, B \leqslant G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Alice: picks a random $a_1 \in A$, $b_1 \in B$ and sends $a_1 b_1$.
- Bob: picks a random $a_2 \in A$, $b_2 \in B$ and sends $a_2 b_2$.
- Common secret: Alice: $b_1 (a_2 b_2) a_1 = a_2 b_1 b_2 a_1 = a_2 a_1 b_1 b_2$.
  Bob: $a_2 (a_1 b_1) b_2$.
- Eve: knows $a_1 a_2$ and $b_1 b_2$, and needs $a_2 a_1 b_1 b_2$.
  This can be done by solving the Factorization Search Problem in A (or B).
  Note that Eve can compute
  $$(a_1 b_1)(a_2 b_2) = a_1 a_2 b_1 b_2 \quad \text{and} \quad (a_2 b_2)(a_1 b_1) = a_2 a_1 b_2 b_1,$$
  but neither of these products equal the secret if $a_1 a_2 \neq a_2 a_1$ and $b_1 b_2 \neq b_2 b_1$.

A protocol based on the Factorization Search Problem

- Public: $G = \langle X \mid R \rangle$ and $A, B \leqslant G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Alice: picks a random $a_1 \in A$, $b_1 \in B$ and sends $a_1 b_1$.
- Bob: picks a random $a_2 \in A$, $b_2 \in B$ and sends $a_2 b_2$.
- Common secret: Alice: $b_1(a_2 b_2)a_1 = a_2 b_1 b_2 a_1 = a_2 a_1 b_1 b_2$.
  Bob: $a_2(a_1 b_1)b_2$.
- Eve: knows $a_1 a_2$ and $b_1 b_2$, and needs $a_2 a_1 b_1 b_2$.
  This can be done by solving the Factorization Search Problem in A (or B).
  Note that Eve can compute
  $$(a_1 b_1)(a_2 b_2) = a_1 a_2 b_1 b_2 \quad \text{and} \quad (a_2 b_2)(a_1 b_1) = a_2 a_1 b_2 b_1,$$
  but neither of these products equal the secret if $a_1 a_2 \neq a_2 a_1$ and $b_1 b_2 \neq b_2 b_1$.

A protocol based on the Factorization Search Problem

- Public: $G = \langle X \mid R \rangle$ and $A, B \leqslant G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Alice: picks a random $a_1 \in A$, $b_1 \in B$ and sends $a_1 b_1$.
- Bob: picks a random $a_2 \in A$, $b_2 \in B$ and sends $a_2 b_2$.
- Common secret: Alice: $b_1(a_2 b_2)a_1 = a_2 b_1 b_2 a_1 = a_2 a_1 b_1 b_2$.
  Bob: $a_2(a_1 b_1)b_2$.
- Eve: knows $a_1 a_2$ and $b_1 b_2$, and needs $a_2 a_1 b_1 b_2$.
  This can be done by solving the Factorization Search Problem in A (or B).
  Note that Eve can compute
  $$(a_1 b_1)(a_2 b_2) = a_1 a_2 b_1 b_2 \quad \text{and} \quad (a_2 b_2)(a_1 b_1) = a_2 a_1 b_2 b_1,$$
  but neither of these products equal the secret if $a_1 a_2 \neq a_2 a_1$ and $b_1 b_2 \neq b_2 b_1$.

A protocol based on the Factorization Search Problem

- Public: $G = \langle X \mid R \rangle$ and $A, B \leqslant G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Alice: picks a random $a_1 \in A$, $b_1 \in B$ and sends $a_1 b_1$.
- Bob: picks a random $a_2 \in A$, $b_2 \in B$ and sends $a_2 b_2$.
- Common secret: Alice: $b_1(a_2 b_2)a_1 = a_2 b_1 b_2 a_1 = a_2 a_1 b_1 b_2$.
  Bob: $a_2(a_1 b_1)b_2$.
- Eve: knows $a_1 a_2$ and $b_1 b_2$, and needs $a_2 a_1 b_1 b_2$.
  This can be done by solving the Factorization Search Problem in A (or B).

  Note that Eve can compute

  $$(a_1 b_1)(a_2 b_2) = a_1 a_2 b_1 b_2 \quad \text{and} \quad (a_2 b_2)(a_1 b_1) = a_2 a_1 b_2 b_1,$$

  but neither of these products equal the secret if $a_1 a_2 \neq a_2 a_1$ and $b_1 b_2 \neq b_2 b_1$.

A protocol based on the Factorization Search Problem

- Public: $G = \langle X \mid R \rangle$ and $A, B \leqslant G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Alice: picks a random $a_1 \in A$, $b_1 \in B$ and sends $a_1 b_1$.
- Bob: picks a random $a_2 \in A$, $b_2 \in B$ and sends $a_2 b_2$.
- Common secret: Alice: $b_1(a_2 b_2)a_1 = a_2 b_1 b_2 a_1 = a_2 a_1 b_1 b_2$.
  Bob: $a_2(a_1 b_1)b_2$.
- Eve: knows $a_1 a_2$ and $b_1 b_2$, and needs $a_2 a_1 b_1 b_2$.
  This can be done by solving the Factorization Search Problem in A (or B).

  Note that Eve can compute

  $$(a_1 b_1)(a_2 b_2) = a_1 a_2 b_1 b_2 \quad \text{and} \quad (a_2 b_2)(a_1 b_1) = a_2 a_1 b_2 b_1,$$

  but neither of these products equal the secret if $a_1 a_2 \neq a_2 a_1$ and $b_1 b_2 \neq b_2 b_1$.

A protocol based on the Factorization Search Problem

- Public: $G = \langle X \mid R \rangle$ and $A, B \leqslant G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Alice: picks a random $a_1 \in A$, $b_1 \in B$ and sends $a_1 b_1$.
- Bob: picks a random $a_2 \in A$, $b_2 \in B$ and sends $a_2 b_2$.
- Common secret: Alice: $b_1(a_2 b_2)a_1 = a_2 b_1 b_2 a_1 = a_2 a_1 b_1 b_2$.
  Bob: $a_2(a_1 b_1)b_2$.
- Eve: knows $a_1 a_2$ and $b_1 b_2$, and needs $a_2 a_1 b_1 b_2$.
  This can be done by solving the Factorization Search Problem in A (or B).

  Note that Eve can compute

  $$(a_1 b_1)(a_2 b_2) = a_1 a_2 b_1 b_2 \quad \text{and} \quad (a_2 b_2)(a_1 b_1) = a_2 a_1 b_2 b_1,$$

  but neither of these products equal the secret if $a_1 a_2 \neq a_2 a_1$ and $b_1 b_2 \neq b_2 b_1$.

A protocol based on the Factorization Search Problem

- Public: $G = \langle X \mid R \rangle$ and $A, B \leqslant G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Alice: picks a random $a_1 \in A$, $b_1 \in B$ and sends $a_1 b_1$.
- Bob: picks a random $a_2 \in A$, $b_2 \in B$ and sends $a_2 b_2$.
- Common secret: Alice: $b_1(a_2 b_2)a_1 = a_2 b_1 b_2 a_1 = a_2 a_1 b_1 b_2$.
  Bob: $a_2(a_1 b_1)b_2$.
- Eve: knows $a_1 a_2$ and $b_1 b_2$, and needs $a_2 a_1 b_1 b_2$.
  This can be done by solving the Factorization Search Problem in A (or B).

  Note that Eve can compute

  $$(a_1 b_1)(a_2 b_2) = a_1 a_2 b_1 b_2 \quad \text{and} \quad (a_2 b_2)(a_1 b_1) = a_2 a_1 b_2 b_1,$$

  but neither of these products equal the secret if $a_1 a_2 \neq a_2 a_1$ and $b_1 b_2 \neq b_2 b_1$.

Kurt's protocol (2006)

- **Public:** $G = \langle X \mid R \rangle$, 10 subgroups $A_1, A_2, A_3, X_1, X_2, B_1, B_2, B_3, Y_1, Y_2 \leqslant G$ such that $[A_2, Y_1] = [A_3, Y_2] = [B_1, X_1] = [B_2, X_2] = 1$.
- **Alice:** picks a random $a_1 \in A_1$, $a_2 \in A_2$, $a_3 \in A_3$, $x_1 \in X_1$, $x_2 \in X_2$, and sends $a_1 x_1$, $x_1^{-1} a_2 x_2$ and $x_2^{-1} a_3$.
- **Bob:** picks a random $b_1 \in B_1$, $b_2 \in B_2$, $b_3 \in B_3$, $y_1 \in Y_1$, $y_2 \in Y_2$, and sends $b_1 y_1$, $y_1^{-1} b_2 y_2$ and $y_2^{-1} b_3$.
- **Common secret:** Alice: $a_1(b_1 y_1) a_2 (y_1^{-1} b_2 y_2) a_3 (y_2^{-1} b_3)$
  Bob: $(a_1 x_1) b_1 (x_1^{-1} a_2 x_2) b_2 (x_2^{-1} a_3) b_3$.
- **Eve:** knows $a_1 x_1$, $x_1^{-1} a_2 x_2$, $x_2^{-1} a_3$, $b_1 y_1$, $y_1^{-1} b_2 y_2$ and $y_2^{-1} b_3$, and needs $a_1 b_1 a_2 b_2 a_3 b_3$.

  This can be done by recovering a_1, a_2, a_3 from $a_1 a_2 a_3 = (a_1 x_1)(x_1^{-1} a_2 x_2)(x_2^{-1} a_3)$, i.e. solving the Triple Factorization Search Problem.

Kurt's protocol (2006)

- Public: $G = \langle X \mid R \rangle$, 10 subgroups $A_1, A_2, A_3, X_1, X_2, B_1, B_2, B_3, Y_1, Y_2 \leqslant G$ such that $[A_2, Y_1] = [A_3, Y_2] = [B_1, X_1] = [B_2, X_2] = 1$.
- Alice: picks a random $a_1 \in A_1$, $a_2 \in A_2$, $a_3 \in A_3$, $x_1 \in X_1$, $x_2 \in X_2$, and sends a_1x_1, $x_1^{-1}a_2x_2$ and $x_2^{-1}a_3$.
- Bob: picks a random $b_1 \in B_1$, $b_2 \in B_2$, $b_3 \in B_3$, $y_1 \in Y_1$, $y_2 \in Y_2$, and sends b_1y_1, $y_1^{-1}b_2y_2$ and $y_2^{-1}b_3$.
- Common secret: Alice: $a_1(b_1y_1)a_2(y_1^{-1}b_2y_2)a_3(y_2^{-1}b_3)$
  Bob: $(a_1x_1)b_1(x_1^{-1}a_2x_2)b_2(x_2^{-1}a_3)b_3$.
- Eve: knows a_1x_1, $x_1^{-1}a_2x_2$, $x_2^{-1}a_3$, b_1y_1, $y_1^{-1}b_2y_2$ and $y_2^{-1}b_3$, and needs $a_1b_1a_2b_2a_3b_3$.

  This can be done by recovering a_1, a_2, a_3 from $a_1a_2a_3 = (a_1x_1)(x_1^{-1}a_2x_2)(x_2^{-1}a_3)$, i.e. solving the Triple Factorization Search Problem.

Kurt's protocol (2006)

- Public: $G = \langle X \mid R \rangle$, 10 subgroups $A_1, A_2, A_3, X_1, X_2, B_1, B_2, B_3, Y_1, Y_2 \leqslant G$ such that $[A_2, Y_1] = [A_3, Y_2] = [B_1, X_1] = [B_2, X_2] = 1$.
- Alice: picks a random $a_1 \in A_1$, $a_2 \in A_2$, $a_3 \in A_3$, $x_1 \in X_1$, $x_2 \in X_2$, and sends $a_1 x_1$, $x_1^{-1} a_2 x_2$ and $x_2^{-1} a_3$.
- Bob: picks a random $b_1 \in B_1$, $b_2 \in B_2$, $b_3 \in B_3$, $y_1 \in Y_1$, $y_2 \in Y_2$, and sends $b_1 y_1$, $y_1^{-1} b_2 y_2$ and $y_2^{-1} b_3$.
- Common secret: Alice: $a_1(b_1 y_1) a_2 (y_1^{-1} b_2 y_2) a_3 (y_2^{-1} b_3)$
  Bob: $(a_1 x_1) b_1 (x_1^{-1} a_2 x_2) b_2 (x_2^{-1} a_3) b_3$.
- Eve: knows $a_1 x_1$, $x_1^{-1} a_2 x_2$, $x_2^{-1} a_3$, $b_1 y_1$, $y_1^{-1} b_2 y_2$ and $y_2^{-1} b_3$, and needs $a_1 b_1 a_2 b_2 a_3 b_3$.

  This can be done by recovering a_1, a_2, a_3 from $a_1 a_2 a_3 = (a_1 x_1)(x_1^{-1} a_2 x_2)(x_2^{-1} a_3)$, i.e. solving the Triple Factorization Search Problem.

Kurt's protocol (2006)

- Public: $G = \langle X \mid R \rangle$, 10 subgroups $A_1, A_2, A_3, X_1, X_2, B_1, B_2, B_3, Y_1, Y_2 \leqslant G$ such that $[A_2, Y_1] = [A_3, Y_2] = [B_1, X_1] = [B_2, X_2] = 1$.
- Alice: picks a random $a_1 \in A_1$, $a_2 \in A_2$, $a_3 \in A_3$, $x_1 \in X_1$, $x_2 \in X_2$, and sends a_1x_1, $x_1^{-1}a_2x_2$ and $x_2^{-1}a_3$.
- Bob: picks a random $b_1 \in B_1$, $b_2 \in B_2$, $b_3 \in B_3$, $y_1 \in Y_1$, $y_2 \in Y_2$, and sends b_1y_1, $y_1^{-1}b_2y_2$ and $y_2^{-1}b_3$.
- Common secret: Alice: $a_1(b_1y_1)a_2(y_1^{-1}b_2y_2)a_3(y_2^{-1}b_3)$
  Bob: $(a_1x_1)b_1(x_1^{-1}a_2x_2)b_2(x_2^{-1}a_3)b_3$.
- Eve: knows a_1x_1, $x_1^{-1}a_2x_2$, $x_2^{-1}a_3$, b_1y_1, $y_1^{-1}b_2y_2$ and $y_2^{-1}b_3$, and needs $a_1b_1a_2b_2a_3b_3$.

  This can be done by recovering a_1, a_2, a_3 from $a_1a_2a_3 = (a_1x_1)(x_1^{-1}a_2x_2)(x_2^{-1}a_3)$, i.e. solving the Triple Factorization Search Problem.

Kurt's protocol (2006)

- Public: $G = \langle X \mid R \rangle$, 10 subgroups $A_1, A_2, A_3, X_1, X_2, B_1, B_2, B_3, Y_1, Y_2 \leqslant G$ such that $[A_2, Y_1] = [A_3, Y_2] = [B_1, X_1] = [B_2, X_2] = 1$.
- Alice: picks a random $a_1 \in A_1$, $a_2 \in A_2$, $a_3 \in A_3$, $x_1 \in X_1$, $x_2 \in X_2$, and sends a_1x_1, $x_1^{-1}a_2x_2$ and $x_2^{-1}a_3$.
- Bob: picks a random $b_1 \in B_1$, $b_2 \in B_2$, $b_3 \in B_3$, $y_1 \in Y_1$, $y_2 \in Y_2$, and sends b_1y_1, $y_1^{-1}b_2y_2$ and $y_2^{-1}b_3$.
- Common secret: Alice: $a_1(b_1y_1)a_2(y_1^{-1}b_2y_2)a_3(y_2^{-1}b_3)$
  Bob: $(a_1x_1)b_1(x_1^{-1}a_2x_2)b_2(x_2^{-1}a_3)b_3$.
- Eve: knows a_1x_1, $x_1^{-1}a_2x_2$, $x_2^{-1}a_3$, b_1y_1, $y_1^{-1}b_2y_2$ and $y_2^{-1}b_3$, and needs $a_1b_1a_2b_2a_3b_3$.

  This can be done by recovering a_1, a_2, a_3 from $a_1a_2a_3 = (a_1x_1)(x_1^{-1}a_2x_2)(x_2^{-1}a_3)$, i.e. solving the Triple Factorization Search Problem.

Kurt's protocol (2006)

- Public: $G = \langle X \mid R \rangle$, 10 subgroups $A_1, A_2, A_3, X_1, X_2, B_1, B_2, B_3, Y_1, Y_2 \leqslant G$ such that $[A_2, Y_1] = [A_3, Y_2] = [B_1, X_1] = [B_2, X_2] = 1$.
- Alice: picks a random $a_1 \in A_1$, $a_2 \in A_2$, $a_3 \in A_3$, $x_1 \in X_1$, $x_2 \in X_2$, and sends $a_1 x_1$, $x_1^{-1} a_2 x_2$ and $x_2^{-1} a_3$.
- Bob: picks a random $b_1 \in B_1$, $b_2 \in B_2$, $b_3 \in B_3$, $y_1 \in Y_1$, $y_2 \in Y_2$, and sends $b_1 y_1$, $y_1^{-1} b_2 y_2$ and $y_2^{-1} b_3$.
- Common secret: Alice: $a_1 (b_1 y_1) a_2 (y_1^{-1} b_2 y_2) a_3 (y_2^{-1} b_3)$
  Bob: $(a_1 x_1) b_1 (x_1^{-1} a_2 x_2) b_2 (x_2^{-1} a_3) b_3$.
- Eve: knows $a_1 x_1$, $x_1^{-1} a_2 x_2$, $x_2^{-1} a_3$, $b_1 y_1$, $y_1^{-1} b_2 y_2$ and $y_2^{-1} b_3$, and needs $a_1 b_1 a_2 b_2 a_3 b_3$.

  This can be done by recovering a_1, a_2, a_3 from $a_1 a_2 a_3 = (a_1 x_1)(x_1^{-1} a_2 x_2)(x_2^{-1} a_3)$, i.e. solving the Triple Factorization Search Problem.

Kurt's protocol (2006)

- Public: $G = \langle X \mid R \rangle$, 10 subgroups $A_1, A_2, A_3, X_1, X_2, B_1, B_2, B_3, Y_1, Y_2 \leqslant G$ such that $[A_2, Y_1] = [A_3, Y_2] = [B_1, X_1] = [B_2, X_2] = 1$.
- Alice: picks a random $a_1 \in A_1$, $a_2 \in A_2$, $a_3 \in A_3$, $x_1 \in X_1$, $x_2 \in X_2$, and sends $a_1 x_1$, $x_1^{-1} a_2 x_2$ and $x_2^{-1} a_3$.
- Bob: picks a random $b_1 \in B_1$, $b_2 \in B_2$, $b_3 \in B_3$, $y_1 \in Y_1$, $y_2 \in Y_2$, and sends $b_1 y_1$, $y_1^{-1} b_2 y_2$ and $y_2^{-1} b_3$.
- Common secret: Alice: $a_1(b_1 y_1) a_2 (y_1^{-1} b_2 y_2) a_3 (y_2^{-1} b_3)$
  Bob: $(a_1 x_1) b_1 (x_1^{-1} a_2 x_2) b_2 (x_2^{-1} a_3) b_3$.
- Eve: knows $a_1 x_1$, $x_1^{-1} a_2 x_2$, $x_2^{-1} a_3$, $b_1 y_1$, $y_1^{-1} b_2 y_2$ and $y_2^{-1} b_3$, and needs $a_1 b_1 a_2 b_2 a_3 b_3$.

  This can be done by recovering a_1, a_2, a_3 from $a_1 a_2 a_3 = (a_1 x_1)(x_1^{-1} a_2 x_2)(x_2^{-1} a_3)$, i.e. solving the Triple Factorization Search Problem.

Stickel's protocol (2005)

- Public: A finite group G, $w \in G$, and $a, b \in G$ with $ab \neq ba$ (of order N and M, respectively).
- Alice: picks a random $0 < n < N$ and $0 < m < M$, and sends $a^n w b^m$.
- Bob: picks a random $0 < n' < N$ and $0 < m' < M$, and sends $a^{n'} w b^{m'}$.
- Common secret: Alice: $a^n(a^{n'} w b^{m'})b^m = a^{n+n'} w b^{m+m'}$
  Bob: $a^{n'}(a^n w b^m)b^{m'} = a^{n+n'} w b^{m+m'}$.
- Eve: knows a, b, $a^n w b^m$ and $a^{n'} w b^{m'}$, and needs $a^{n+n'} w b^{m+m'}$.

  This can be done by solving a variation of the Discrete Logarithm Problem (in G).

  Or... finding alternative $x, y \in G$ such that $xa = ax$, $yb = by$ and $xwy = a^n w b^m$. Then,
  $x(a^{n'} w b^{m'})y = a^{n'} xwy b^{m'} = a^{n'}(a^n w b^m) b^{m'} = a^{n+n'} w b^{m+m'}$.

Stickel's protocol (2005)

- Public: A finite group G, $w \in G$, and $a, b \in G$ with $ab \neq ba$ (of order N and M, respectively).
- Alice: picks a random $0 < n < N$ and $0 < m < M$, and sends $a^n wb^m$.
- Bob: picks a random $0 < n' < N$ and $0 < m' < M$, and sends $a^{n'} wb^{m'}$.
- Common secret: Alice: $a^n(a^{n'} wb^{m'})b^m = a^{n+n'} wb^{m+m'}$
  Bob: $a^{n'}(a^n wb^m)b^{m'} = a^{n+n'} wb^{m+m'}$.
- Eve: knows a, b, $a^n wb^m$ and $a^{n'} wb^{m'}$, and needs $a^{n+n'} wb^{m+m'}$.

  This can be done by solving a variation of the Discrete Logarithm Problem (in G).

  Or... finding alternative $x, y \in G$ such that $xa = ax$, $yb = by$ and $xwy = a^n wb^m$. Then,
  $x(a^{n'} wb^{m'})y = a^{n'} xwyb^{m'} = a^{n'}(a^n wb^m)b^{m'} = a^{n+n'} wb^{m+m'}$.

Stickel's protocol (2005)

- Public: A finite group G, $w \in G$, and $a, b \in G$ with $ab \neq ba$ (of order N and M, respectively).
- Alice: picks a random $0 < n < N$ and $0 < m < M$, and sends $a^n w b^m$.
- Bob: picks a random $0 < n' < N$ and $0 < m' < M$, and sends $a^{n'} w b^{m'}$.
- Common secret: Alice: $a^n(a^{n'} w b^{m'}) b^m = a^{n+n'} w b^{m+m'}$
  Bob: $a^{n'}(a^n w b^m) b^{m'} = a^{n+n'} w b^{m+m'}$.
- Eve: knows a, b, $a^n w b^m$ and $a^{n'} w b^{m'}$, and needs $a^{n+n'} w b^{m+m'}$.

  This can be done by solving a variation of the Discrete Logarithm Problem (in G).

  Or... finding alternative $x, y \in G$ such that $xa = ax$, $yb = by$ and $xwy = a^n w b^m$. Then,
  $x(a^{n'} w b^{m'})y = a^{n'} xwy b^{m'} = a^{n'}(a^n w b^m) b^{m'} = a^{n+n'} w b^{m+m'}$.

Stickel's protocol (2005)

- Public: A finite group G, $w \in G$, and $a, b \in G$ with $ab \neq ba$ (of order N and M, respectively).
- Alice: picks a random $0 < n < N$ and $0 < m < M$, and sends $a^n w b^m$.
- Bob: picks a random $0 < n' < N$ and $0 < m' < M$, and sends $a^{n'} w b^{m'}$.
- Common secret: Alice: $a^n (a^{n'} w b^{m'}) b^m = a^{n+n'} w b^{m+m'}$
  Bob: $a^{n'} (a^n w b^m) b^{m'} = a^{n+n'} w b^{m+m'}$.
- Eve: knows a, b, $a^n w b^m$ and $a^{n'} w b^{m'}$, and needs $a^{n+n'} w b^{m+m'}$.

  This can be done by solving a variation of the Discrete Logarithm Problem (in G).

  Or... finding alternative $x, y \in G$ such that $xa = ax$, $yb = by$ and $xwy = a^n w b^m$. Then,
  $x(a^{n'} w b^{m'})y = a^{n'} xwy b^{m'} = a^{n'} (a^n w b^m) b^{m'} = a^{n+n'} w b^{m+m'}$.

Stickel's protocol (2005)

- Public: A finite group G, $w \in G$, and $a, b \in G$ with $ab \neq ba$ (of order N and M, respectively).
- Alice: picks a random $0 < n < N$ and $0 < m < M$, and sends $a^n w b^m$.
- Bob: picks a random $0 < n' < N$ and $0 < m' < M$, and sends $a^{n'} w b^{m'}$.
- Common secret: Alice: $a^n(a^{n'} w b^{m'}) b^m = a^{n+n'} w b^{m+m'}$
  Bob: $a^{n'}(a^n w b^m) b^{m'} = a^{n+n'} w b^{m+m'}$.
- Eve: knows a, b, $a^n w b^m$ and $a^{n'} w b^{m'}$, and needs $a^{n+n'} w b^{m+m'}$.

  This can be done by solving a variation of the Discrete Logarithm Problem (in G).

  Or... finding alternative $x, y \in G$ such that $xa = ax$, $yb = by$ and $xwy = a^n w b^m$. Then,
  $x(a^{n'} w b^{m'})y = a^{n'} xwy b^{m'} = a^{n'}(a^n w b^m) b^{m'} = a^{n+n'} w b^{m+m'}$.

Stickel's protocol (2005)

- Public: A finite group G, $w \in G$, and $a, b \in G$ with $ab \neq ba$ (of order N and M, respectively).
- Alice: picks a random $0 < n < N$ and $0 < m < M$, and sends $a^n w b^m$.
- Bob: picks a random $0 < n' < N$ and $0 < m' < M$, and sends $a^{n'} w b^{m'}$.
- Common secret:
$$\begin{aligned} \text{Alice:} \quad & a^n(a^{n'} w b^{m'})b^m = a^{n+n'} w b^{m+m'} \\ \text{Bob:} \quad & a^{n'}(a^n w b^m)b^{m'} = a^{n+n'} w b^{m+m'}. \end{aligned}$$
- Eve: knows a, b, $a^n w b^m$ and $a^{n'} w b^{m'}$, and needs $a^{n+n'} w b^{m+m'}$.

  This can be done by solving a variation of the Discrete Logarithm Problem (in G).

  Or... finding alternative $x, y \in G$ such that $xa = ax$, $yb = by$ and $xwy = a^n w b^m$. Then,
  $x(a^{n'} w b^{m'})y = a^{n'} xwy b^{m'} = a^{n'}(a^n w b^m) b^{m'} = a^{n+n'} w b^{m+m'}$.

Stickel's protocol (2005)

- Public: A finite group G, $w \in G$, and $a, b \in G$ with $ab \neq ba$ (of order N and M, respectively).
- Alice: picks a random $0 < n < N$ and $0 < m < M$, and sends $a^n wb^m$.
- Bob: picks a random $0 < n' < N$ and $0 < m' < M$, and sends $a^{n'} wb^{m'}$.
- Common secret: Alice: $a^n(a^{n'} wb^{m'})b^m = a^{n+n'} wb^{m+m'}$
  Bob: $a^{n'}(a^n wb^m)b^{m'} = a^{n+n'} wb^{m+m'}$.
- Eve: knows a, b, $a^n wb^m$ and $a^{n'} wb^{m'}$, and needs $a^{n+n'} wb^{m+m'}$.

  This can be done by solving a variation of the Discrete Logarithm Problem (in G).

  Or... finding alternative $x, y \in G$ such that $xa = ax$, $yb = by$ and $xwy = a^n wb^m$. Then,
  $x(a^{n'} wb^{m'})y = a^{n'} xwyb^{m'} = a^{n'}(a^n wb^m)b^{m'} = a^{n+n'} wb^{m+m'}$.

Outline

Anshel-Anshel-Goldfeld protocol (1999)

This is a protocol genuinely based on non-commutativity (i.e. without using any commuting subgroups).

- Public: A group $G = \langle X \mid R \rangle$ and elements $a_1, \ldots, a_m \in G$, $b_1, \ldots, b_n \in G$.
- Alice: picks a word $x = x(a_1, \ldots, a_m)$, and sends $b_1^x, \ldots, b_n^x$.
- Bob: picks a word $y = y(b_1, \ldots, b_n)$, and sends $a_1^y, \ldots, a_m^y$.
- Common secret:

$$\begin{array}{rl} \text{Alice:} & x(a_1^y, \ldots, a_m^y) = x^y = y^{-1}xy, \text{ and } x^{-1}(y^{-1}xy) = [x, y] \\ \text{Bob:} & y(b_1^x, \ldots, b_n^x) = y^x = x^{-1}yx, \text{ and } (x^{-1}yx)^{-1}y = [x, y]. \end{array}$$

- Eve: knows $a_1, \ldots, a_m$, $b_1, \ldots, b_n$, $a_1^y, \ldots, a_m^y$, $b_1^x, \ldots, b_n^x$ and needs $[x, y]$.

  This can be done by solving the Multiple Restricted Search Conjugacy Problem.

  But there are subtleties here...

Anshel-Anshel-Goldfeld protocol (1999)

This is a protocol genuinely based on non-commutativity (i.e. without using any commuting subgroups).

- Public: A group $G = \langle X \mid R \rangle$ and elements $a_1, \ldots, a_m \in G$, $b_1, \ldots, b_n \in G$.
- Alice: picks a word $x = x(a_1, \ldots, a_m)$, and sends $b_1^x, \ldots, b_n^x$.
- Bob: picks a word $y = y(b_1, \ldots, b_n)$, and sends $a_1^y, \ldots, a_m^y$.
- Common secret:

  Alice: $x(a_1^y, \ldots, a_m^y) = x^y = y^{-1}xy$, and $x^{-1}(y^{-1}xy) = [x, y]$

  Bob: $y(b_1^x, \ldots, b_n^x) = y^x = x^{-1}yx$, and $(x^{-1}yx)^{-1}y = [x, y]$.
- Eve: knows $a_1, \ldots, a_m$, $b_1, \ldots, b_n$, $a_1^y, \ldots, a_m^y$, $b_1^x, \ldots, b_n^x$ and needs $[x, y]$.

  This can be done by solving the Multiple Restricted Search Conjugacy Problem.

  But there are subtleties here...

Anshel-Anshel-Goldfeld protocol (1999)

This is a protocol genuinely based on non-commutativity (i.e. without using any commuting subgroups).

- Public: A group $G = \langle X \mid R \rangle$ and elements $a_1, \ldots, a_m \in G$, $b_1, \ldots, b_n \in G$.
- Alice: picks a word $x = x(a_1, \ldots, a_m)$, and sends $b_1^x, \ldots, b_n^x$.
- Bob: picks a word $y = y(b_1, \ldots, b_n)$, and sends $a_1^y, \ldots, a_m^y$.
- Common secret:

  Alice: $x(a_1^y, \ldots, a_m^y) = x^y = y^{-1}xy$, and $x^{-1}(y^{-1}xy) = [x, y]$

  Bob: $y(b_1^x, \ldots, b_n^x) = y^x = x^{-1}yx$, and $(x^{-1}yx)^{-1}y = [x, y]$.
- Eve: knows $a_1, \ldots, a_m$, $b_1, \ldots, b_n$, $a_1^y, \ldots, a_m^y$, $b_1^x, \ldots, b_n^x$ and needs $[x, y]$.

  This can be done by solving the Multiple Restricted Search Conjugacy Problem.

  But there are subtleties here...

Anshel-Anshel-Goldfeld protocol (1999)

This is a protocol genuinely based on non-commutativity (i.e. without using any commuting subgroups).

- Public: A group $G = \langle X \mid R \rangle$ and elements $a_1, \ldots, a_m \in G$, $b_1, \ldots, b_n \in G$.
- Alice: picks a word $x = x(a_1, \ldots, a_m)$, and sends $b_1^x, \ldots, b_n^x$.
- Bob: picks a word $y = y(b_1, \ldots, b_n)$, and sends $a_1^y, \ldots, a_m^y$.
- Common secret:

  Alice: $x(a_1^y, \ldots, a_m^y) = x^y = y^{-1}xy$, and $x^{-1}(y^{-1}xy) = [x, y]$

  Bob: $y(b_1^x, \ldots, b_n^x) = y^x = x^{-1}yx$, and $(x^{-1}yx)^{-1}y = [x, y]$.
- Eve: knows $a_1, \ldots, a_m$, $b_1, \ldots, b_n$, $a_1^y, \ldots, a_m^y$, $b_1^x, \ldots, b_n^x$ and needs $[x, y]$.

  This can be done by solving the Multiple Restricted Search Conjugacy Problem.

  But there are subtleties here...

Anshel-Anshel-Goldfeld protocol (1999)

This is a protocol genuinely based on non-commutativity (i.e. without using any commuting subgroups).

- Public: A group $G = \langle X \mid R \rangle$ and elements $a_1, \ldots, a_m \in G$, $b_1, \ldots, b_n \in G$.
- Alice: picks a word $x = x(a_1, \ldots, a_m)$, and sends $b_1^x, \ldots, b_n^x$.
- Bob: picks a word $y = y(b_1, \ldots, b_n)$, and sends $a_1^y, \ldots, a_m^y$.
- Common secret:

$$\begin{array}{rl} \text{Alice:} & x(a_1^y, \ldots, a_m^y) = x^y = y^{-1}xy, \text{ and } x^{-1}(y^{-1}xy) = [x, y] \\ \text{Bob:} & y(b_1^x, \ldots, b_n^x) = y^x = x^{-1}yx, \text{ and } (x^{-1}yx)^{-1}y = [x, y]. \end{array}$$

- Eve: knows $a_1, \ldots, a_m$, $b_1, \ldots, b_n$, $a_1^y, \ldots, a_m^y$, $b_1^x, \ldots, b_n^x$ and needs $[x, y]$.

  This can be done by solving the Multiple Restricted Search Conjugacy Problem.

  But there are subtleties here...

Anshel-Anshel-Goldfeld protocol (1999)

This is a protocol genuinely based on non-commutativity (i.e. without using any commuting subgroups).

- Public: A group $G = \langle X \mid R \rangle$ and elements $a_1, \ldots, a_m \in G$, $b_1, \ldots, b_n \in G$.
- Alice: picks a word $x = x(a_1, \ldots, a_m)$, and sends $b_1^x, \ldots, b_n^x$.
- Bob: picks a word $y = y(b_1, \ldots, b_n)$, and sends $a_1^y, \ldots, a_m^y$.
- Common secret:

  Alice: $x(a_1^y, \ldots, a_m^y) = x^y = y^{-1}xy$, and $x^{-1}(y^{-1}xy) = [x, y]$

  Bob: $y(b_1^x, \ldots, b_n^x) = y^x = x^{-1}yx$, and $(x^{-1}yx)^{-1}y = [x, y]$.
- Eve: knows $a_1, \ldots, a_m$, $b_1, \ldots, b_n$, $a_1^y, \ldots, a_m^y$, $b_1^x, \ldots, b_n^x$ and needs $[x, y]$.

  This can be done by solving the Multiple Restricted Search Conjugacy Problem.

  But there are subtleties here...

Anshel-Anshel-Goldfeld protocol (1999)

This is a protocol genuinely based on non-commutativity (i.e. without using any commuting subgroups).

- Public: A group $G = \langle X \mid R \rangle$ and elements $a_1, \ldots, a_m \in G$, $b_1, \ldots, b_n \in G$.
- Alice: picks a word $x = x(a_1, \ldots, a_m)$, and sends $b_1^x, \ldots, b_n^x$.
- Bob: picks a word $y = y(b_1, \ldots, b_n)$, and sends $a_1^y, \ldots, a_m^y$.
- Common secret:
  Alice: $x(a_1^y, \ldots, a_m^y) = x^y = y^{-1}xy$, and $x^{-1}(y^{-1}xy) = [x, y]$
  Bob: $y(b_1^x, \ldots, b_n^x) = y^x = x^{-1}yx$, and $(x^{-1}yx)^{-1}y = [x, y]$.
- Eve: knows $a_1, \ldots, a_m$, $b_1, \ldots, b_n$, $a_1^y, \ldots, a_m^y$, $b_1^x, \ldots, b_n^x$ and needs $[x, y]$.

  This can be done by solving the Multiple Restricted Search Conjugacy Problem.

  But there are subtleties here...

Anshel-Anshel-Goldfeld protocol (1999)

This is a protocol genuinely based on non-commutativity (i.e. without using any commuting subgroups).

- Public: A group $G = \langle X \mid R \rangle$ and elements $a_1, \ldots, a_m \in G$, $b_1, \ldots, b_n \in G$.
- Alice: picks a word $x = x(a_1, \ldots, a_m)$, and sends $b_1^x, \ldots, b_n^x$.
- Bob: picks a word $y = y(b_1, \ldots, b_n)$, and sends $a_1^y, \ldots, a_m^y$.
- Common secret:

  $$\begin{array}{rl} \text{Alice:} & x(a_1^y, \ldots, a_m^y) = x^y = y^{-1}xy, \ \text{and } x^{-1}(y^{-1}xy) = [x, y] \\ \text{Bob:} & y(b_1^x, \ldots, b_n^x) = y^x = x^{-1}yx, \ \text{and } (x^{-1}yx)^{-1}y = [x, y]. \end{array}$$

- Eve: knows $a_1, \ldots, a_m$, $b_1, \ldots, b_n$, $a_1^y, \ldots, a_m^y$, $b_1^x, \ldots, b_n^x$ and needs $[x, y]$.

  This can be done by solving the Multiple Restricted Search Conjugacy Problem.

  But there are subtleties here...

Anshel-Anshel-Goldfeld protocol (1999)

- The element x conjugating $b_1, \ldots, b_n$ into $b_1^x, \ldots, b_n^x$ need not be unique.
- After solving the Multiple Search Conjugacy Problem, Eve will find $x' = c_b x$ where $c_b \in C_G(b_1) \cap \cdots \cap C_G(b_n)$, $y' = c_a y$ where $c_a \in C_G(a_1) \cap \cdots \cap C_G(a_m)$.
- Now, $[x', y'] = [x, y] \iff c_a$ commutes with c_b:

$$[x', y'] = (x^{-1}c_b^{-1})(y^{-1}c_a^{-1})(c_b x)(c_a y) = x^{-1}y^{-1}c_b^{-1}c_a^{-1}c_b c_a xy.$$

- The only visible way to ensure this is to have $x' \in A$ (so $c_b \in A$ and $[c_a, c_b] = 1$), or $y' \in B$.
- Hence, the (unrestricted) Multiple Search Conjugacy Problem does not seem to be enough in order to break the system.

Anshel-Anshel-Goldfeld protocol (1999)

- The element x conjugating $b_1, \ldots, b_n$ into $b_1^x, \ldots, b_n^x$ need not be unique.
- After solving the Multiple Search Conjugacy Problem, Eve will find $x' = c_b x$ where $c_b \in C_G(b_1) \cap \cdots \cap C_G(b_n)$,
  $y' = c_a y$ where $c_a \in C_G(a_1) \cap \cdots \cap C_G(a_m)$.
- Now, $[x', y'] = [x, y] \iff c_a$ commutes with c_b:

$$[x', y'] = (x^{-1}c_b^{-1})(y^{-1}c_a^{-1})(c_b x)(c_a y) = x^{-1}y^{-1}c_b^{-1}c_a^{-1}c_b c_a xy.$$

- The only visible way to ensure this is to have $x' \in A$ (so $c_b \in A$ and $[c_a, c_b] = 1$), or $y' \in B$.
- Hence, the (unrestricted) Multiple Search Conjugacy Problem does not seem to be enough in order to break the system.

Anshel-Anshel-Goldfeld protocol (1999)

- The element x conjugating $b_1, \ldots, b_n$ into $b_1^x, \ldots, b_n^x$ need not be unique.
- After solving the Multiple Search Conjugacy Problem, Eve will find $x' = c_b x$ where $c_b \in C_G(b_1) \cap \cdots \cap C_G(b_n)$, $y' = c_a y$ where $c_a \in C_G(a_1) \cap \cdots \cap C_G(a_m)$.
- Now, $[x', y'] = [x, y] \Leftrightarrow c_a$ commutes with c_b:

$$[x', y'] = (x^{-1} c_b^{-1})(y^{-1} c_a^{-1})(c_b x)(c_a y) = x^{-1} y^{-1} c_b^{-1} c_a^{-1} c_b c_a x y.$$

- The only visible way to ensure this is to have $x' \in A$ (so $c_b \in A$ and $[c_a, c_b] = 1$), or $y' \in B$.
- Hence, the (unrestricted) Multiple Search Conjugacy Problem does not seem to be enough in order to break the system.

Anshel-Anshel-Goldfeld protocol (1999)

- The element x conjugating $b_1, \ldots, b_n$ into $b_1^x, \ldots, b_n^x$ need not be unique.
- After solving the Multiple Search Conjugacy Problem, Eve will find $x' = c_b x$ where $c_b \in C_G(b_1) \cap \cdots \cap C_G(b_n)$,
  $y' = c_a y$ where $c_a \in C_G(a_1) \cap \cdots \cap C_G(a_m)$.
- Now, $[x', y'] = [x, y] \Leftrightarrow c_a$ commutes with c_b:

$$[x', y'] = (x^{-1} c_b^{-1})(y^{-1} c_a^{-1})(c_b x)(c_a y) = x^{-1} y^{-1} c_b^{-1} c_a^{-1} c_b c_a x y.$$

- The only visible way to ensure this is to have $x' \in A$ (so $c_b \in A$ and $[c_a, c_b] = 1$), or $y' \in B$.
- Hence, the (unrestricted) Multiple Search Conjugacy Problem does not seem to be enough in order to break the system.

Anshel-Anshel-Goldfeld protocol (1999)

- The element x conjugating $b_1, \ldots, b_n$ into $b_1^x, \ldots, b_n^x$ need not be unique.
- After solving the Multiple Search Conjugacy Problem, Eve will find
$$\begin{aligned} x' &= c_b x \quad \text{where } c_b \in C_G(b_1) \cap \cdots \cap C_G(b_n), \\ y' &= c_a y \quad \text{where } c_a \in C_G(a_1) \cap \cdots \cap C_G(a_m). \end{aligned}$$
- Now, $[x', y'] = [x, y] \;\Leftrightarrow\; c_a$ commutes with c_b:
$$[x', y'] = (x^{-1}c_b^{-1})(y^{-1}c_a^{-1})(c_b x)(c_a y) = x^{-1}y^{-1}c_b^{-1}c_a^{-1}c_b c_a xy.$$
- The only visible way to ensure this is to have $x' \in A$ (so $c_b \in A$ and $[c_a, c_b] = 1$), or $y' \in B$.
- Hence, the (unrestricted) Multiple Search Conjugacy Problem does not seem to be enough in order to break the system.

Outline

Authentication protocols

- These are protocols to ensure that somebody is really who is claiming to be.

- General setting: Every player has a public name, and a secret key. When I call somebody by his name, he must provide me a proof that he knows the corresponding secret key (so, he is who is supposed to be), but without revealing any information about the key itself.

- Many key establishment protocols can be modified to become authentication protocols.

Authentication protocols

- These are protocols to ensure that somebody is really who is claiming to be.
- General setting: Every player has a public name, and a secret key. When I call somebody by his name, he must provide me a proof that he knows the corresponding secret key (so, he is who is supposed to be), but without revealing any information about the key itself.
- Many key establishment protocols can be modified to become authentication protocols.

Authentication protocols

- These are protocols to ensure that somebody is really who is claiming to be.
- General setting: Every player has a public name, and a secret key. When I call somebody by his name, he must provide me a proof that he knows the corresponding secret key (so, he is who is supposed to be), but without revealing any information about the key itself.
- Many key establishment protocols can be modified to become authentication protocols.

Diffie-Hellman authentication protocol

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Every player has a secret key $a \in \mathbb{N}$, and public name $g^a \mod p$.
- Bob, the *verifier*, wants to be sure that Alice (say, Ms. "$g^a \mod p$"), the *prover*, is who is supposed to be.
- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$ (a *challenge*).
- Alice: sends $(g^b)^a \mod p$.
- Bob: verifies whether $(g^b)^a = (g^a)^b \mod p$.
- Eve: knows p, g and $g^a \mod p$, and needs a to be able to impersonate Alice. This is the Discrete Logarithm Problem.

Diffie-Hellman authentication protocol

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Every player has a secret key $a \in \mathbb{N}$, and public name $g^a \mod p$.

- Bob, the *verifier*, wants to be sure that Alice (say, Ms. "$g^a \mod p$"), the *prover*, is who is supposed to be.

- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$ (a *challenge*).
- Alice: sends $(g^b)^a \mod p$.
- Bob: verifies whether $(g^b)^a = (g^a)^b \mod p$.

- Eve: knows p, g and $g^a \mod p$, and needs a to be able to impersonate Alice. This is the Discrete Logarithm Problem.

Diffie-Hellman authentication protocol

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Every player has a secret key $a \in \mathbb{N}$, and public name $g^a \mod p$.

- Bob, the *verifier*, wants to be sure that Alice (say, Ms. "$g^a \mod p$"), the *prover*, is who is supposed to be.

- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$ (a *challenge*).
- Alice: sends $(g^b)^a \mod p$.
- Bob: verifies whether $(g^b)^a = (g^a)^b \mod p$.

- Eve: knows p, g and $g^a \mod p$, and needs a to be able to impersonate Alice. This is the Discrete Logarithm Problem.

Diffie-Hellman authentication protocol

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Every player has a secret key $a \in \mathbb{N}$, and public name $g^a \mod p$.

- Bob, the *verifier*, wants to be sure that Alice (say, Ms. "$g^a \mod p$"), the *prover*, is who is supposed to be.

- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$ (a *challenge*).
- Alice: sends $(g^b)^a \mod p$.
- Bob: verifies whether $(g^b)^a = (g^a)^b \mod p$.

- Eve: knows p, g and $g^a \mod p$, and needs a to be able to impersonate Alice. This is the Discrete Logarithm Problem.

Diffie-Hellman authentication protocol

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Every player has a secret key $a \in \mathbb{N}$, and public name $g^a \mod p$.

- Bob, the *verifier*, wants to be sure that Alice (say, Ms. "$g^a \mod p$"), the *prover*, is who is supposed to be.

- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$ (a *challenge*).
- Alice: sends $(g^b)^a \mod p$.
- Bob: verifies whether $(g^b)^a = (g^a)^b \mod p$.

- Eve: knows p, g and $g^a \mod p$, and needs a to be able to impersonate Alice. This is the Discrete Logarithm Problem.

Diffie-Hellman authentication protocol

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Every player has a secret key $a \in \mathbb{N}$, and public name $g^a \mod p$.

- Bob, the *verifier*, wants to be sure that Alice (say, Ms. "$g^a \mod p$"), the *prover*, is who is supposed to be.

- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$ (a *challenge*).
- Alice: sends $(g^b)^a \mod p$.
- Bob: verifies whether $(g^b)^a = (g^a)^b \mod p$.

- Eve: knows p, g and $g^a \mod p$, and needs a to be able to impersonate Alice. This is the Discrete Logarithm Problem.

Diffie-Hellman authentication protocol

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Every player has a secret key $a \in \mathbb{N}$, and public name $g^a \mod p$.

- Bob, the *verifier*, wants to be sure that Alice (say, Ms. "$g^a \mod p$"), the *prover*, is who is supposed to be.

- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$ (a *challenge*).
- Alice: sends $(g^b)^a \mod p$.
- Bob: verifies whether $(g^b)^a = (g^a)^b \mod p$.

- Eve: knows p, g and $g^a \mod p$, and needs a to be able to impersonate Alice. This is the Discrete Logarithm Problem.

Diffie-Hellman authentication protocol

- Public: p (prime) and $g \notin p\mathbb{Z}$.
- Every player has a secret key $a \in \mathbb{N}$, and public name $g^a \mod p$.

- Bob, the *verifier*, wants to be sure that Alice (say, Ms. "$g^a \mod p$"), the *prover*, is who is supposed to be.

- Bob: picks a random $b \in \mathbb{N}$, and sends $g^b \mod p$ (a *challenge*).
- Alice: sends $(g^b)^a \mod p$.
- Bob: verifies whether $(g^b)^a = (g^a)^b \mod p$.

- Eve: knows p, g and $g^a \mod p$, and needs a to be able to impersonate Alice. This is the Discrete Logarithm Problem.

Diffie-Hellman-like authentication protocol

- Public: $G = \langle X \mid R \rangle$ and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.
- Bob: picks a random $b \in B$, and sends $u^b = b^{-1}ub$.
- Alice: sends $(u^b)^a = u^{ba}$.
- Bob: verifies whether $u^{ba} = (u^a)^b$.
- Eve: knows u and u^a, and needs a to be able to authenticate as Alice to Bob. This is the Discrete Logarithm Problem.

Diffie-Hellman-like authentication protocol

- Public: $G = \langle X \mid R \rangle$ and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).

- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

- Bob: picks a random $b \in B$, and sends $u^b = b^{-1}ub$.
- Alice: sends $(u^b)^a = u^{ba}$.
- Bob: verifies whether $u^{ba} = (u^a)^b$.

- Eve: knows u and u^a, and needs a to be able to authenticate as Alice to Bob. This is the Discrete Logarithm Problem.

Diffie-Hellman-like authentication protocol

- Public: $G = \langle X \mid R \rangle$ and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).

- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

- Bob: picks a random $b \in B$, and sends $u^b = b^{-1}ub$.
- Alice: sends $(u^b)^a = u^{ba}$.
- Bob: verifies whether $u^{ba} = (u^a)^b$.

- Eve: knows u and u^a, and needs a to be able to authenticate as Alice to Bob. This is the Discrete Logarithm Problem.

Diffie-Hellman-like authentication protocol

- Public: $G = \langle X \mid R \rangle$ and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.
- Bob: picks a random $b \in B$, and sends $u^b = b^{-1}ub$.
- Alice: sends $(u^b)^a = u^{ba}$.
- Bob: verifies whether $u^{ba} = (u^a)^b$.
- Eve: knows u and u^a, and needs a to be able to authenticate as Alice to Bob. This is the Discrete Logarithm Problem.

Diffie-Hellman-like authentication protocol

- Public: $G = \langle X \mid R \rangle$ and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).

- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

- Bob: picks a random $b \in B$, and sends $u^b = b^{-1}ub$.
- Alice: sends $(u^b)^a = u^{ba}$.
- Bob: verifies whether $u^{ba} = (u^a)^b$.

- Eve: knows u and u^a, and needs a to be able to authenticate as Alice to Bob. This is the Discrete Logarithm Problem.

Diffie-Hellman-like authentication protocol

- Public: $G = \langle X \mid R \rangle$ and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).

- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

- Bob: picks a random $b \in B$, and sends $u^b = b^{-1}ub$.
- Alice: sends $(u^b)^a = u^{ba}$.
- Bob: verifies whether $u^{ba} = (u^a)^b$.

- Eve: knows u and u^a, and needs a to be able to authenticate as Alice to Bob. This is the Discrete Logarithm Problem.

Diffie-Hellman-like authentication protocol

- Public: $G = \langle X \mid R \rangle$ and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).

- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

- Bob: picks a random $b \in B$, and sends $u^b = b^{-1}ub$.
- Alice: sends $(u^b)^a = u^{ba}$.
- Bob: verifies whether $u^{ba} = (u^a)^b$.

- Eve: knows u and u^a, and needs a to be able to authenticate as Alice to Bob. This is the Discrete Logarithm Problem.

Diffie-Hellman-like authentication protocol

- Public: $G = \langle X \mid R \rangle$ and $A, B \subseteq G$ such that $[a, b] = 1$ $\forall a \in A$, $\forall b \in B$.
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).

- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

- Bob: picks a random $b \in B$, and sends $u^b = b^{-1}ub$.
- Alice: sends $(u^b)^a = u^{ba}$.
- Bob: verifies whether $u^{ba} = (u^a)^b$.

- Eve: knows u and u^a, and needs a to be able to authenticate as Alice to Bob. This is the Discrete Logarithm Problem.

Sibert-Dehornoy-Girault authentication protocol (2006)

- Public: $G = \langle X \mid R \rangle$ (and no commuting subgroups!).
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

First (wrong) attempt:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$, and $y = b$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$.
- Eve: can easily impersonate Alice, by acting in the same way (a plays no role).

Sibert-Dehornoy-Girault authentication protocol (2006)

- Public: $G = \langle X \mid R \rangle$ (and no commuting subgroups!).
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

First (wrong) attempt:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$, and $y = b$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$.
- Eve: can easily impersonate Alice, by acting in the same way (a plays no role).

Sibert-Dehornoy-Girault authentication protocol (2006)

- Public: $G = \langle X \mid R \rangle$ (and no commuting subgroups!).
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

First (wrong) attempt:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$, and $y = b$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$.
- Eve: can easily impersonate Alice, by acting in the same way (a plays no role).

Sibert-Dehornoy-Girault authentication protocol (2006)

- Public: $G = \langle X \mid R \rangle$ (and no commuting subgroups!).
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

First (wrong) attempt:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$, and $y = b$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$.
- Eve: can easily impersonate Alice, by acting in the same way (a plays no role).

Sibert-Dehornoy-Girault authentication protocol (2006)

- Public: $G = \langle X \mid R \rangle$ (and no commuting subgroups!).
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

First (wrong) attempt:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$, and $y = b$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$.
- Eve: can easily impersonate Alice, by acting in the same way (a plays no role).

Sibert-Dehornoy-Girault authentication protocol (2006)

- Public: $G = \langle X \mid R \rangle$ (and no commuting subgroups!).
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

First (wrong) attempt:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$, and $y = b$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$.
- Eve: can easily impersonate Alice, by acting in the same way (a plays no role).

Sibert-Dehornoy-Girault authentication protocol (2006)

- Public: $G = \langle X \mid R \rangle$ (and no commuting subgroups!).
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

Second (wrong) attempt:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$, and $z = ab$.
- Bob: verifies whether $z^{-1} \cdot u \cdot z = x$.
- Eve: can easily impersonate Alice: choosing $b \in B$ and sending $x = b^{-1}ub$ and $z = b$ will cheat Bob.

Sibert-Dehornoy-Girault authentication protocol (2006)

- Public: $G = \langle X \mid R \rangle$ (and no commuting subgroups!).
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

Second (wrong) attempt:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$, and $z = ab$.
- Bob: verifies whether $z^{-1} \cdot u \cdot z = x$.
- Eve: can easily impersonate Alice: choosing $b \in B$ and sending $x = b^{-1}ub$ and $z = b$ will cheat Bob.

Sibert-Dehornoy-Girault authentication protocol (2006)

- Public: $G = \langle X \mid R \rangle$ (and no commuting subgroups!).
- Every player has a secret key $a \in A$, and public name (u, u^a), where $u \in G$ is arbitrary (and $u^a = a^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^a)") is who is supposed to be.

Second (wrong) attempt:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$, and $z = ab$.
- Bob: verifies whether $z^{-1} \cdot u \cdot z = x$.
- Eve: can easily impersonate Alice: choosing $b \in B$ and sending $x = b^{-1}ub$ and $z = b$ will cheat Bob.

Sibert-Dehornoy-Girault authentication protocol (2006)

But combining both, it works:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$ (the *commitment*).
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$ and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$ (if $\alpha = 0$) or whether $z^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^a, i.e. solving the Conjugacy Search Problem.

Sibert-Dehornoy-Girault authentication protocol (2006)

But combining both, it works:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$ (the *commitment*).
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$ and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$ (if $\alpha = 0$) or whether $z^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^a, i.e. solving the Conjugacy Search Problem.

Sibert-Dehornoy-Girault authentication protocol (2006)

But combining both, it works:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$ (the *commitment*).
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$ and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$ (if $\alpha = 0$) or whether $z^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^a, i.e. solving the Conjugacy Search Problem.

Sibert-Dehornoy-Girault authentication protocol (2006)

But combining both, it works:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$ (the *commitment*).
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$ and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$ (if $\alpha = 0$) or whether $z^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^a, i.e. solving the Conjugacy Search Problem.

Sibert-Dehornoy-Girault authentication protocol (2006)

But combining both, it works:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$ (the *commitment*).
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$ and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$ (if $\alpha = 0$) or whether $z^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^a, i.e. solving the Conjugacy Search Problem.

Sibert-Dehornoy-Girault authentication protocol (2006)

But combining both, it works:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$ (the *commitment*).
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$ and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$ (if $\alpha = 0$) or whether $z^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^a, i.e. solving the Conjugacy Search Problem.

Sibert-Dehornoy-Girault authentication protocol (2006)

But combining both, it works:

- Alice: picks a random $b \in B$, and sends $x = b^{-1}(u^a)b$ (the *commitment*).
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$ and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $y^{-1} \cdot u^a \cdot y = x$ (if $\alpha = 0$) or whether $z^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^a, i.e. solving the Conjugacy Search Problem.

The Twisted Conjugacy Problem

One can use the same idea, but replacing the Conjugacy Search Problem to the harder Twisted Conjugacy Search Problem.

- Twisted Conjugacy Problem: "given $u, v \in G$ and $\varphi \colon G \to G$, decide whether $v =_G (x\varphi)^{-1}ux$ for some $x \in G$".

Solv. Twisted Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Twisted Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Twisted Conjugacy Search Problem: "given $u, v \in G$, $\varphi \colon G \to G$, and the information that u and v are φ-twisted conjugated to each other in G, find an $x \in G$ such that $v =_G (x\varphi)^{-1}ux$".

TCSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The Twisted Conjugacy Problem

One can use the same idea, but replacing the Conjugacy Search Problem to the harder Twisted Conjugacy Search Problem.

- Twisted Conjugacy Problem: "given $u, v \in G$ and $\varphi \colon G \to G$, decide whether $v =_G (x\varphi)^{-1}ux$ for some $x \in G$".

Solv. Twisted Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Twisted Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Twisted Conjugacy Search Problem: "given $u, v \in G$, $\varphi \colon G \to G$, and the information that u and v are φ-twisted conjugated to each other in G, find an $x \in G$ such that $v =_G (x\varphi)^{-1}ux$".

TCSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The Twisted Conjugacy Problem

One can use the same idea, but replacing the Conjugacy Search Problem to the harder Twisted Conjugacy Search Problem.

- Twisted Conjugacy Problem: "given $u, v \in G$ and $\varphi\colon G \to G$, decide whether $v =_G (x\varphi)^{-1}ux$ for some $x \in G$".

Solv. Twisted Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Twisted Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Twisted Conjugacy Search Problem: "given $u, v \in G$, $\varphi\colon G \to G$, and the information that u and v are φ-twisted conjugated to each other in G, find an $x \in G$ such that $v =_G (x\varphi)^{-1}ux$".

TCSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The Twisted Conjugacy Problem

One can use the same idea, but replacing the Conjugacy Search Problem to the harder Twisted Conjugacy Search Problem.

- Twisted Conjugacy Problem: "given $u, v \in G$ and $\varphi \colon G \to G$, decide whether $v =_G (x\varphi)^{-1}ux$ for some $x \in G$".

Solv. Twisted Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Twisted Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Twisted Conjugacy Search Problem: "given $u, v \in G$, $\varphi \colon G \to G$, and the information that u and v are φ-twisted conjugated to each other in G, find an $x \in G$ such that $v =_G (x\varphi)^{-1}ux$".

TCSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

The Twisted Conjugacy Problem

One can use the same idea, but replacing the Conjugacy Search Problem to the harder Twisted Conjugacy Search Problem.

- Twisted Conjugacy Problem: “given $u, v \in G$ and $\varphi\colon G \to G$, decide whether $v =_G (x\varphi)^{-1}ux$ for some $x \in G$”.

Solv. Twisted Conjugacy Problem $\Longrightarrow$ solv. Conjugacy Problem.

Solv. Twisted Conjugacy Problem $\not\Longleftarrow$ solv. Conjugacy Problem.

- Twisted Conjugacy Search Problem: “given $u, v \in G$, $\varphi\colon G \to G$, and the information that u and v are φ-twisted conjugated to each other in G, find an $x \in G$ such that $v =_G (x\varphi)^{-1}ux$”.

TCSP is always solvable (brute force searching over all possible $x \in G$), but at which complexity this is a much more delicate question.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi \colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1} ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi \colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1} ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi\colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi \colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1} ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi \colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1} ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi \colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1} ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi \colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi \colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1} ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi\colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

Shpilrain-Ushakov authentication protocol (2008)

- Public: $G = \langle X \mid R \rangle$ and $\varphi\colon G \to G$, an endomorphism.
- Every player has a secret key $a \in A$, and public name (u, u^{a_φ}), where $u \in G$ is arbitrary (and $u^{a_\varphi} = (a\varphi)^{-1}ua$).
- Bob wants to be sure that Alice (say, Ms. "(u, u^{a_φ})") is who is supposed to be.
- Alice: picks a random $b \in B$, and sends the commitment $x = (b\varphi)^{-1}(u^{a_\varphi})b = (b\varphi)^{-1}(a\varphi)^{-1}uab = ((ab)\varphi)^{-1}u(ab)$.
- Bob: picks and sends a random bit $\alpha = 0, 1$.
- Alice: sends $y = b$ if $\alpha = 0$, and $z = ab$ if $\alpha = 1$.
- Bob: verifies whether $(y\varphi)^{-1} \cdot u^{a_\varphi} \cdot y = x$ (if $\alpha = 0$) or whether $(z\varphi)^{-1} \cdot u \cdot z = x$ (if $\alpha = 1$).
- Repeat these last three steps, k times.

- Eve: has to send the commitment before knowing the future values of α; so, acting like before, she only has probability $\frac{1}{2^k}$ to succeed.
- Eve's alternative is finding a from u and u^{a_φ}, i.e. solving the Twisted Conjugacy Search Problem.

THANKS