An Efficient CDH-based Signature Scheme With a Tight Security Reduction

Benoit Chevallier-Mames[1,2]

[1]Gemplus ARSC/STD/CSE
[2]Ecole Normale Supérieure, Paris

— Crypto '05 —

Contents

Signature Scheme

Signature Scheme Definition

A signature scheme $\textsc{Sig} = (\textsc{GenKey}, \textsc{Sign}, \textsc{Verify})$ is defined by the three following algorithms:

Signature Scheme

Signature Scheme Definition

A signature scheme $\textsc{Sig} = (\textsc{GenKey}, \textsc{Sign}, \textsc{Verify})$ is defined by the three following algorithms:

- The *key generation algorithm* $\textsc{GenKey}$.

Signature Scheme

Signature Scheme Definition

A signature scheme $\textsc{Sig} = (\textsc{GenKey}, \textsc{Sign}, \textsc{Verify})$ is defined by the three following algorithms:

- The *key generation algorithm* $\textsc{GenKey}$.
- The *signing algorithm* $\textsc{Sign}$.

Signature Scheme

Signature Scheme Definition

A signature scheme $\textsc{Sig} = (\textsc{GenKey}, \textsc{Sign}, \textsc{Verify})$ is defined by the three following algorithms:

- The *key generation algorithm* $\textsc{GenKey}$.
- The *signing algorithm* $\textsc{Sign}$.
- The *verification algorithm* $\textsc{Verify}$.

Proving Security
The attacker model

Goal of the adversary for a Signature Scheme

- Total break of the scheme (recovering the private key) – BK

Proving Security

The attacker model

Goal of the adversary for a Signature Scheme

- Total break of the scheme (recovering the private key) – BK
- Universal forgery (can sign any message) – UF

Proving Security
The attacker model

Goal of the adversary for a Signature Scheme

- Total break of the scheme (recovering the private key) – BK
- Universal forgery (can sign any message) – UF
- Existential forgery (can sign one message) – EUF

Proving Security
The attacker model

Goal of the adversary for a Signature Scheme

- Total break of the scheme (recovering the private key) – BK
- Universal forgery (can sign any message) – UF
- Existential forgery (can sign one message) – EUF

Information available to the Attacker

- No message attack – NMA

Proving Security
The attacker model

Goal of the adversary for a Signature Scheme

- Total break of the scheme (recovering the private key) – BK
- Universal forgery (can sign any message) – UF
- Existential forgery (can sign one message) – EUF

Information available to the Attacker

- No message attack – NMA
- Known message attack – KMA

Proving Security
The attacker model

Goal of the adversary for a Signature Scheme

- Total break of the scheme (recovering the private key) – BK
- Universal forgery (can sign any message) – UF
- Existential forgery (can sign one message) – EUF

Information available to the Attacker

- No message attack – NMA
- Known message attack – KMA
- Chosen message attack – CMA

Proving Security
The attacker model

Goal of the adversary for a Signature Scheme

- Total break of the scheme (recovering the private key) – BK
- Universal forgery (can sign any message) – UF
- Existential forgery (can sign one message) – EUF

Information available to the Attacker

- No message attack – NMA
- Known message attack – KMA
- Chosen message attack – CMA

Then, the strongest model is EUF-CMA.

Proving Security

Reductionist Security

Reduction to hard problems

An attacker that breaks the signature scheme is transformed into a solver of one hard problem.

Proving Security
Reductionist Security

Reduction to hard problems

An attacker that breaks the signature scheme is transformed into a solver of one hard problem.

- e-th root problem (*a.k.a.* RSA problem)

Proving Security
Reductionist Security

Reduction to hard problems

An attacker that breaks the signature scheme is transformed into a solver of one hard problem.

- e-th root problem (*a.k.a.* RSA problem)
- Factorization

Proving Security
Reductionist Security

Reduction to hard problems

An attacker that breaks the signature scheme is transformed into a solver of one hard problem.

- e-th root problem (*a.k.a.* RSA problem)
- Factorization
- Computational Diffie Hellman problem (being given g^x and g^a, find g^{ax}) – CDH

Proving Security
Reductionist Security

Reduction to hard problems

An attacker that breaks the signature scheme is transformed into a solver of one hard problem.

- e-th root problem (*a.k.a.* RSA problem)
- Factorization
- Computational Diffie Hellman problem (being given g^x and g^a, find g^{ax}) – CDH
- Discrete Logarithm (being given g^x and g, find x) – DL

Proving Security
Reductionist Security

Reduction to hard problems

An attacker that breaks the signature scheme is transformed into a solver of one hard problem.

- e-th root problem (*a.k.a.* RSA problem)
- Factorization
- Computational Diffie Hellman problem (being given g^x and g^a, find g^{ax}) – CDH
- Discrete Logarithm (being given g^x and g, find x) – DL

Tightness of the Reduction

An attacker that breaks the signature scheme with probability ε and within time τ is transformed into a solver of one hard problem, with probability ε' and within time τ'.

Proving Security
Reductionist Security

Reduction to hard problems

An attacker that breaks the signature scheme is transformed into a solver of one hard problem.

- e-th root problem (*a.k.a.* RSA problem)
- Factorization
- Computational Diffie Hellman problem (being given g^x and g^a, find g^{ax}) – CDH
- Discrete Logarithm (being given g^x and g, find x) – DL

Tightness of the Reduction

An attacker that breaks the signature scheme with probability ε and within time τ is transformed into a solver of one hard problem, with probability ε' and within time τ'.

- The reduction is *loose* if $\frac{\tau'}{\varepsilon'} \ll \frac{\tau}{\varepsilon}$

Proving Security
Reductionist Security

Reduction to hard problems

An attacker that breaks the signature scheme is transformed into a solver of one hard problem.

- *e*-th root problem (*a.k.a.* RSA problem)
- Factorization
- Computational Diffie Hellman problem (being given g^x and g^a, find g^{ax}) – CDH
- Discrete Logarithm (being given g^x and g, find x) – DL

Tightness of the Reduction

An attacker that breaks the signature scheme with probability ε and within time τ is transformed into a solver of one hard problem, with probability ε' and within time τ'.

- The reduction is *loose* if $\frac{\tau'}{\varepsilon'} \ll \frac{\tau}{\varepsilon}$
- The reduction is *tight* if $\frac{\tau'}{\varepsilon'} \sim \frac{\tau}{\varepsilon}$

The *EDL* signature scheme

It is independently proposed in [CP92],[JS99] and proved in [GJ03] is defined as follows.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

THE *EDL* SIGNATURE SCHEME

It is independently proposed in [CP92],[JS99] and proved in [GJ03] is defined as follows.

KEY GENERATION: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

SIGNATURE: To sign a message $m \in \mathcal{M}$, one first randomly chooses $r \in \{0,1\}^{\ell_r}$ and $k \in \mathbb{Z}_q$, then

1. $u = g^k$ (can be computed online)
2. $h = \mathcal{H}(m, r)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, r, s, c)$.

The *EDL* signature scheme

It is independently proposed in [CP92],[JS99] and proved in [GJ03] is defined as follows.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Signature: To sign a message $m \in \mathcal{M}$, one first randomly chooses $r \in \{0,1\}^{\ell_r}$ and $k \in \mathbb{Z}_q$, then

1. $u = g^k$ (can be computed online)
2. $h = \mathcal{H}(m, r)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, r, s, c)$.

Verification: To verify a signature $\sigma = (z, r, s, c)$ on a message m, one computes $h' = \mathcal{H}(m, r)$, $u' = g^s y^{-c}$ and $v' = h'^s z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(g, h', y, z, u', v')$.

The *EDL* signature scheme

It is independently proposed in [CP92],[JS99] and proved in [GJ03] is defined as follows.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Signature: To sign a message $m \in \mathcal{M}$, one first randomly chooses $r \in \{0,1\}^{\ell_r}$ and $k \in \mathbb{Z}_q$, then

1. $u = g^k$ (can be computed online)
2. $h = \mathcal{H}(m, r)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, r, s, c)$.

Verification: To verify a signature $\sigma = (z, r, s, c)$ on a message m, one computes $h' = \mathcal{H}(m, r)$, $u' = g^s y^{-c}$ and $v' = h'^s z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(g, h', y, z, u', v')$.

Correctness:

The *EDL* signature scheme

It is independently proposed in [CP92],[JS99] and proved in [GJ03] is defined as follows.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Signature: To sign a message $m \in \mathcal{M}$, one first randomly chooses $r \in \{0,1\}^{\ell_r}$ and $k \in \mathbb{Z}_q$, then

1. $u = g^k$ (can be computed online)
2. $h = \mathcal{H}(m, r)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, r, s, c)$.

Verification: To verify a signature $\sigma = (z, r, s, c)$ on a message m, one computes $h' = \mathcal{H}(m, r)$, $u' = g^s y^{-c}$ and $v' = h'^s z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(g, h', y, z, u', v')$.

Correctness: $h' = \mathcal{H}(m, r) = h$

The *EDL* signature scheme

It is independently proposed in [CP92],[JS99] and proved in [GJ03] is defined as follows.

KEY GENERATION: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

SIGNATURE: To sign a message $m \in \mathcal{M}$, one first randomly chooses $r \in \{0,1\}^{\ell_r}$ and $k \in \mathbb{Z}_q$, then

1. $u = g^k$ (can be computed online)
2. $h = \mathcal{H}(m, r)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, r, s, c)$.

VERIFICATION: To verify a signature $\sigma = (z, r, s, c)$ on a message m, one computes $h' = \mathcal{H}(m, r)$, $u' = g^s y^{-c}$ and $v' = h'^s z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(g, h', y, z, u', v')$.

CORRECTNESS: $u' = g^s y^{-c} = g^{k+cx} y^{-c} = g^{k+cx} g^{-cx} = g^k = u$

The *EDL* signature scheme

It is independently proposed in [CP92],[JS99] and proved in [GJ03] is defined as follows.

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Signature: To sign a message $m \in \mathcal{M}$, one first randomly chooses $r \in \{0,1\}^{\ell_r}$ and $k \in \mathbb{Z}_q$, then

1. $u = g^k$ (can be computed online)
2. $h = \mathcal{H}(m, r)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, r, s, c)$.

Verification: To verify a signature $\sigma = (z, r, s, c)$ on a message m, one computes $h' = \mathcal{H}(m, r)$, $u' = g^s\, y^{-c}$ and $v' = h'^s\, z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(g, h', y, z, u', v')$.

Correctness: $v' = h'^s\, z^{-c} = h^{k+cx}\, z^{-c} = h^{k+cx}\, h^{-cx} = h^k = v$

The *EDL* signature scheme

It is independently proposed in [CP92],[JS99] and proved in [GJ03] is defined as follows.

KEY GENERATION: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

SIGNATURE: To sign a message $m \in \mathcal{M}$, one first randomly chooses $r \in \{0,1\}^{\ell_r}$ and $k \in \mathbb{Z}_q$, then

1. $u = g^k$ (can be computed online)
2. $h = \mathcal{H}(m, r)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, r, s, c)$.

VERIFICATION: To verify a signature $\sigma = (z, r, s, c)$ on a message m, one computes $h' = \mathcal{H}(m, r)$, $u' = g^s y^{-c}$ and $v' = h'^s z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(g, h', y, z, u', v')$.

CORRECTNESS: So $c = \mathcal{G}(g, h', y, z, u', v')$

The Security of EDL

Security of EDL

The scheme is extremely secure:

The Security of EDL

Security of EDL

The scheme is extremely secure:

- Attacker model: EUF-CMA.

The Security of EDL

Security of EDL

The scheme is extremely secure:

- Attacker model: EUF-CMA.
- Hard problem: Computational Diffie Hellman

The Security of EDL

Security of EDL

The scheme is extremely secure:

- Attacker model: EUF-CMA.
- Hard problem: Computational Diffie Hellman
- The reduction is tight, in the *random oracle model*

Features of EDL

EDL:

- Tight reduction to the CDH problem in the random oracle model

Features of EDL

EDL:

- Tight reduction to the CDH problem in the random oracle model
- Short keys, short group

Features of EDL

EDL:

- Tight reduction to the CDH problem in the random oracle model
- Short keys, short group
- Signature size is $\ell_p + 2\ell_q + \ell_r$, which is for subgroup of $\mathbb{Z}_p$: $1024 + 2 * 176 + 111 = 1487$ bits, and for elliptic curve groups: $3 * 176 + 111 = 639$ bits

Features of EDL

EDL:

- Tight reduction to the CDH problem in the random oracle model
- Short keys, short group
- Signature size is $\ell_p + 2\ell_q + \ell_r$, which is for subgroup of $\mathbb{Z}_p$: $1024 + 2 * 176 + 111 = 1487$ bits, and for elliptic curve groups: $3 * 176 + 111 = 639$ bits
- No online possibility (or [ST01] technique, that makes signature longer and cost more time to sign and verify)

Other variants of EDL

Other variants of *EDL*

- Katz-Wang scheme ([KW03]), based on the Decisional Diffie-Hellman (DDH)

Other variants of EDL

Other variants of *EDL*

- Katz-Wang scheme ([KW03]), based on the Decisional Diffie-Hellman (DDH)
- Katz-Wang scheme ([KW03]), based on the CDH, with shorter signatures

Our Scheme

EDL is defined as follows:

KEY GENERATION: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

SIGNATURE: To sign a message $m \in \mathcal{M}$, one first randomly chooses $r \in \{0,1\}^{\ell_r}$ and $k \in \mathbb{Z}_q$, then

1. $u = g^k$
2. $h = \mathcal{H}(m, r)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, r, s, c)$.

VERIFICATION: To verify a signature $\sigma = (z, r, s, c)$ on a message m, one computes $h' = \mathcal{H}(m, r)$, $u' = g^s\, y^{-c}$ and $v' = h'^s\, z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(g, h', y, z, u', v')$.

Our Scheme

Step 1 of our construction is defined as follows (Appendix B):

KEY GENERATION: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

SIGNATURE: To sign a message $m \in \mathcal{M}$, one first randomly chooses $k \in \mathbb{Z}_q$, then

1. $u = g^k$
2. $h = \mathcal{H}(m, u)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, s, c)$.

VERIFICATION: To verify a signature $\sigma = (z, s, c)$ on a message m, one computes $h' = \mathcal{H}(m, u)$, $u' = g^s y^{-c}$ and $v' = h'^s z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(g, h', y, z, u', v')$.

Our Scheme

Our scheme is defined as follows (Section 4):

Key generation: The private key is a random number $x \in \mathbb{Z}_q$. The corresponding public key is $y = g^x$.

Signature: To sign a message $m \in \mathcal{M}$, one first randomly chooses $k \in \mathbb{Z}_q$, then

1. $u = g^k$
2. $h = \mathcal{H}(u)$
3. $z = h^x$
4. $v = h^k$
5. $c = \mathcal{G}(m, g, h, y, z, u, v)$
6. $s = k + cx \bmod q$

The signature on m is $\sigma = (z, s, c)$.

Verification: To verify a signature $\sigma = (z, s, c)$ on a message m, one computes $h' = \mathcal{H}(u)$, $u' = g^s\, y^{-c}$ and $v' = h'^s\, z^{-c}$. The signature σ is accepted iff $c = \mathcal{G}(m, g, h', y, z, u', v')$.

Features of Our Scheme

Our Scheme:

- Tight reduction to the CDH problem in the random oracle model

Features of Our Scheme

Our Scheme:

- Tight reduction to the CDH problem in the random oracle model
- Short keys, short group

Features of Our Scheme

Our Scheme:

- Tight reduction to the CDH problem in the random oracle model
- Short keys, short group
- Signature size is $\ell_p + 2\ell_q$, which is for subgroup of $\mathbb{Z}_p$: $1024 + 2 * 176 = 1376$ bits (-7%), and for elliptic curve groups: $3 * 176 = 528$ bits (-17%)

Features of Our Scheme

Our Scheme:

- Tight reduction to the CDH problem in the random oracle model
- Short keys, short group
- Signature size is $\ell_p + 2\ell_q$, which is for subgroup of $\mathbb{Z}_p$: $1024 + 2 * 176 = 1376$ bits (-7%), and for elliptic curve groups: $3 * 176 = 528$ bits (-17%)
- Online possibility

Exact Security of Our Scheme

We have the following theorem:

Theorem

Let $\mathcal{A}$ be an adversary which can produce, with success probability ε, an existential forgery under a chosen-message attack within time τ, after q_h queries to the hash oracles and q_s queries to the signing oracle, in the random oracle model. Then the computational Diffie-Hellman problem can be solved with success probability ε' within time τ', with

$$\varepsilon' \geq \varepsilon - 2q_s\left(\frac{q_s + q_h}{q}\right)$$

and

$$\tau' \lesssim \tau + (6q_s + q_h)\tau_0$$

where τ_0 is the time for an exponentiation in $G_{g,q}$.

Intuition of the Proof of Security

Imagine a forger returns a forge $(\hat{z}, \hat{s}, \hat{c})$, we compute corresponding $\hat{u}$, $\hat{v}$. As in *EDL*, we write $\hat{u} = g^k$, $\hat{v} = \hat{h}^{k'}$ and $\hat{z} = \hat{h}^{x'}$ (we do not know k, k', x, x').

Intuition of the Proof of Security

Imagine a forger returns a forge $(\hat{z}, \hat{s}, \hat{c})$, we compute corresponding $\hat{u}$, $\hat{v}$. As in *EDL*, we write $\hat{u} = g^k$, $\hat{v} = \hat{h}^{k'}$ and $\hat{z} = \hat{h}^{x'}$ (we do not know k, k', x, x').

As the signature is valid,

1. $u' = g^s y^{-c}$
2. $v' = h'^s z^{-c}$

So, in the exponent world,

1. $k = \hat{s} - \hat{c}x \bmod q$
2. $k' = \hat{s} - \hat{c}x' \bmod q$

Intuition of the Proof of Security

Imagine a forger returns a forge $(\hat{z}, \hat{s}, \hat{c})$, we compute corresponding $\hat{u}$, $\hat{v}$. As in *EDL*, we write $\hat{u} = g^k$, $\hat{v} = \hat{h}^{k'}$ and $\hat{z} = \hat{h}^{x'}$ (we do not know k, k', x, x').

As the signature is valid,

1. $u' = g^s y^{-c}$
2. $v' = h'^s z^{-c}$

So, in the exponent world,

1. $k = \hat{s} - \hat{c}x \bmod q$
2. $k' = \hat{s} - \hat{c}x' \bmod q$

Then, if $x \neq x'$, we have $\hat{c} = \mathcal{G}(\hat{m}, g, \hat{h}, y, \hat{h}^{x'}, g^k, \hat{h}^{k'}) = \frac{k-k'}{x'-x} \bmod q$.

Intuition of the Proof of Security

Imagine a forger returns a forge $(\hat{z}, \hat{s}, \hat{c})$, we compute corresponding $\hat{u}$, $\hat{v}$. As in *EDL*, we write $\hat{u} = g^k$, $\hat{v} = \hat{h}^{k'}$ and $\hat{z} = \hat{h}^{x'}$ (we do not know k, k', x, x').

As the signature is valid,

1. $u' = g^s y^{-c}$
2. $v' = h'^s z^{-c}$

So, in the exponent world,

1. $k = \hat{s} - \hat{c}x \bmod q$
2. $k' = \hat{s} - \hat{c}x' \bmod q$

Then, if $x \neq x'$, we have $\hat{c} = \mathcal{G}(\hat{m}, g, \hat{h}, y, \hat{h}^{x'}, g^k, \hat{h}^{k'}) = \frac{k-k'}{x'-x} \bmod q$.

This is impossible to find with a probability $\frac{q_{\mathcal{G}}}{q}$. Apart this negligible error, we know that $x = x'$ (btw, $k = k'$), and so that $\hat{z} = \hat{h}^x$.

Intuition of the Proof of Security

Imagine a forger returns a forge $(\hat{z}, \hat{s}, \hat{c})$, we compute corresponding $\hat{u}$, $\hat{v}$. As in *EDL*, we write $\hat{u} = g^k$, $\hat{v} = \hat{h}^{k'}$ and $\hat{z} = \hat{h}^{x'}$ (we do not know k, k', x, x').

As the signature is valid,

1. $u' = g^s y^{-c}$
2. $v' = h'^s z^{-c}$

So, in the exponent world,

1. $k = \hat{s} - \hat{c}x \bmod q$
2. $k' = \hat{s} - \hat{c}x' \bmod q$

Then, if $x \neq x'$, we have $\hat{c} = \mathcal{G}(\hat{m}, g, \hat{h}, y, \hat{h}^{x'}, g^k, \hat{h}^{k'}) = \frac{k-k'}{x'-x} \bmod q$.

This is impossible to find with a probability $\frac{q_{\mathcal{G}}}{q}$. Apart this negligible error, we know that $x = x'$ (btw, $k = k'$), and so that $\hat{z} = \hat{h}^x$.

Conclusion:

- the forger is able to find a new h and its corresponding h^x
- or the forger is able to reuse an h that was given by the simulator/actual signer

Intuition of the Proof of Security

Two Cases:

- the forger is able to find a new h and its corresponding h^x
- or the forger is able to reuse an h that was given by the simulator/actual signer

Intuition of the Proof of Security

Two Cases:

- the forger is able to find a new h and its corresponding h^x
- or the forger is able to reuse an h that was given by the simulator/actual signer

In case 1, the proof shows that the attacker can be used to solve a CDH (g, g^a, g^x): roughly, the simulator returns to hash queries $h = (g^a)^d$, for a random d. Then, he deduces the answer of the CDH challenge $\hat{z}^{1/d} = \hat{h}^{x/d} = ((g^a)^d)^{x/d} = g^{ax}$.

Intuition of the Proof of Security

Two Cases:

- the forger is able to find a new h and its corresponding h^x
- or the forger is able to reuse an h that was given by the simulator/actual signer

In case 1, the proof shows that the attacker can be used to solve a CDH (g, g^a, g^x): roughly, the simulator returns to hash queries $h = (g^a)^d$, for a random d. Then, he deduces the answer of the CDH challenge $\hat{z}^{1/d} = \hat{h}^{x/d} = ((g^a)^d)^{x/d} = g^{ax}$.

In case 2, the proof shows that the attacker can be used to solve a DL (or collision on $\mathcal{H}$ or $\mathcal{G}$ hash functions). As $h = \mathcal{H}(u) = \hat{h} = \mathcal{H}(\hat{u})$, $u = \hat{u}$. So $u = g^s\, y^{-c} = \hat{u} = g^{\hat{s}}\, y^{-\hat{c}}$. If $c \neq \hat{c}$, we recover the DL as $x = \frac{s-\hat{s}}{c-\hat{c}} \bmod q$.

Conclusion

Conclusion:

- More details in the paper, or in its full version, at http://eprint.iacr.org/2005/035

Conclusion

Conclusion:

- More details in the paper, or in its full version, at `http://eprint.iacr.org/2005/035`
- Thank you