Design Validations for Discrete Logarithm Based Signature Schemes

Marcel Šebek

Department of Algebra
Faculty of Mathematics and Physics
Charles University in Prague

29. 11. 2010 / Fall School of Algebra

Outline

1. Definitions
   - Basic Definitions
   - DSA-like Signature Scheme Instances
   - Generalized Signature Schemes
2. Security Results

Signature Scheme

Definition (Signature Scheme)

Signature scheme is a tuple (**Key-Gen**, **Sign**, **Ver**) of polynomial time probabilistic algorithms.

- **Key-Gen** generates a pair of keys (**Priv**, **Pub**)
- **Sign** takes **Priv** and a message M and produces a signature S
- **Ver** takes a signature S and a public key **Pub** and checks whether the signature is valid

Adaptively Chosen Message Attack Unforgeability

Definition (Existential Unforgeability)

The signature scheme is *unforgeable* if no adversary who is given the public key and the signatures of n messages adaptively chosen by himself, can produce the signature of a new message with non-negligible probability.

Remark

This is the strongest known form of the signature scheme unforgeability.

Hash Functions and their Properties

Definition

A *hash function* is a function $H : 2^* \to 2^k$ for some fixed k.

Definition

Assume that domain of a hash function H is finite. Then H is *ℓ-collision-free* if there is no ℓ-tuple $(x_1, \ldots, x_\ell)$ of pairwise distinct elements for which $H(x_1) = \cdots = H(x_\ell)$. If $\ell = 2$, we just say *collision-free*.

Definition

A hash function H is *ℓ-collision-resistant* if the above mentioned ℓ-tuple is computationally impossible to be found. If $\ell = 2$, we just say *collision-resistant*.

Random Oracle

- A model of an ideal hash function
- Returns random (uniformly distributed) value for any new input
- Returns consistent answers for previous inputs
- A simulator may define uniformly distributed answers for inputs that have not yet been queried
- Proofs in random oracle models are questionable

Common Assumptions for the Signature Schemes

- Let p, q be primes, $q \mid p - 1$
- Let $g \in \mathbb{Z}_p^*$ be an element of order q
- Let G and H be hash functions with ranges $\mathcal{G}$ and $\mathcal{H}$
  - $q/2 < |\mathcal{G}|$
  - $|\mathcal{H}| < 2q$
  - G is ℓ-collision resistant or ℓ-collision free
  - H is a random oracle

The **Key-Gen** algorithm

- Take the private key $X \in \mathbb{Z}_q^*$ at random
- Compute the public key $Y = g^X \mod p$

DSA

The **Sign** algorithm

- Take $k \in \mathbb{Z}_q^*$ at random
- Compute the following:
  $R = g^k \mod p \quad T = R \mod q$
  $U = H(M) \qquad S = (U + XT)/k \mod q$
- The signature is (S, T)

The **Ver** algorithm

Check whether $g^{U/S} Y^{T/S} \mod p \mod q = T$, where $U = H(M)$

DSA-I Variant

The **Sign** algorithm

- Take $k \in \mathbb{Z}_q^*$ at random
- Compute the following:
  $R = g^k \mod p \quad T = G(R)$
  $U = H(M) \qquad S = (U + XT)/k \mod q$
- The signature is (S, T)

The **Ver** algorithm

Check whether $g^{U/S} Y^{T/S} \mod p \mod q = T$, where $U = H(M)$

DSA-II Variant

The **Sign** algorithm

- Take $k \in \mathbb{Z}_q^*$ at random
- Compute the following:
  $R = g^k \mod p \quad T = G(R)$
  $U = H(M, T) \quad S = (U + XT)/k \mod q$
- The signature is (S, T)

The **Ver** algorithm

Check whether $g^{U/S} Y^{T/S} \mod p \mod q = T$, where $U = H(M, T)$

KCDSA

The **Sign** algorithm

- Take $k \in \mathbb{Z}_q^*$ at random
- Compute the following:
  $R = g^k \mod p \quad T = G(M)$
  $U = H(R) \qquad S = (k - T \oplus U)/X \mod q$
- The signature is (S, U)

The **Ver** algorithm

- Compute $T = G(M)$, $E_G = T \oplus U$, $W = g^{E_G} Y^S \mod p$
- Check whether $U = H(W)$

Common Assumptions for TEGTSS Schemes

F_i Functions

There exist functions

$$F_1 : (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{G}, \mathcal{H}) \to \mathbb{Z}_q$$
$$F_2 : (\mathbb{Z}_q, \mathcal{G}, \mathcal{H}) \to \mathbb{Z}_q$$
$$F_3 : (\mathbb{Z}_q, \mathcal{G}, \mathcal{H}) \to \mathbb{Z}_q$$

satisfying for all $(k, X, T, U) \in (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{G}, \mathcal{H})$ the following equation

$$F_2\left(F_1\left(k, X, T, U\right), T, U\right) + X \cdot F_3\left(F_1\left(k, X, T, U\right), T, U\right) = k \mod q$$

Common Assumptions for TEGTSS Schemes

Verification Equation

Definition

A tuple (W, S, T, U) satisfies the TEGTSS Verification Equation if $W = g^{E_G} Y^{E_Y} \mod p$ for $E_G = F_2(S, T, U)$ and $E_Y = F_3(S, T, U)$.

Note

If the tuple (W, S, T, U) satisfies the TEGTSS Verification Equation, it does not necessarily mean that $U = H(W)$ for TEGTSS-I or $T = G(W)$ for TEGTSS-II (i.e. the tuple is verifiable).

TEGTSS-I
Algorithms

The **Sign** algorithm

- Take $k \in \mathbb{Z}_q^*$ at random
- Compute the following:
  $R = g^k \mod p \quad T = G(M)$
  $U = H(R) \quad S = F_1(k, X, T, U)$
- The signature is (S, U)

The **Ver** algorithm

- Compute $T = G(M)$, $E_G = F_2(S, T, U)$, $E_Y = F_3(S, T, U)$ and $W = g^{E_G} Y^{E_Y} \mod p$
- Check whether $U = H(W)$

TEGTSS-I
Required Properties

Let the tuples (W, S_i, T_i, U_i), $i = 1, 2$ satisfy the TEGTSS Verification equation, let (W, S_1, T_1, U_1) be verifiable. Then the following must hold:

- if $T_1 \neq T_2$ and (W, S_2, T_2, U_2) is verifiable, then $F_3(S_1, T_1, U_1) \neq F_3(S_2, T_2, U_2)$
- let (W, S_1, T_1, U_1) be fixed, then there is a one-to-one mapping between the values U_2 and T_2 such that $F_3(S_1, T_1, U_1) = F_3(S_2, T_2, U_2)$

TEGTSS-I
Example

Lemma

The KCDSA scheme is a TEGTSS-I scheme.

Proof.

Trivial and technical. □

TEGTSS-II
Algorithms

The **Sign** algorithm

- Take $k \in \mathbb{Z}_q^*$ at random
- Compute the following:
  $R = g^k \mod p \quad T = G(R)$
  $U = H(M, T) \quad S = F_1(k, X, T, U)$
- The signature is (S, T)

The **Ver** algorithm

- Compute $U = H(M, T)$, $E_G = F_2(S, T, U)$, $E_Y = F_3(S, T, U)$ and $W = g^{E_G} Y^{E_Y} \mod p$
- Check whether $T = G(W)$

TEGTSS-II
Required Properties

For given T, E_G and E_Y, there exists a unique pair (U, S) such that $E_G = F_2(S, T, U)$ and $E_Y = F_3(S, T, U)$. This pair is easy to find.

TEGTSS-II
Example

Lemma

The DSA-II scheme is a TEGTSS-II scheme.

Proof.

Trivial and technical. □

Main Result

Theorem

Let us have an attacker $\mathcal{A}$ that is able to perform an existential forgery against TEGTSS scheme with probability $\epsilon > 4/q$ after Q queries to the H function. Let H be a random oracle.

- *for TEGTSS-I scheme, if G is collision-resistant, then one extracts the secret key X with less than $24Q/\epsilon$ replays of $\mathcal{A}$, with constant probability greater than $1/100$*
- *for TEGTSS-II scheme, if*
  - *G is $(\ell+1)$-collision-resistant*
  - *or, $x \mapsto G(g^x \mod p)$ is $(\ell+1)$-collision-free,*

  then one extracts the secret key X with less than $25Q\log_2(2\ell)/\epsilon$ replays of $\mathcal{A}$, with constant probability greater than $1/100$.

Proof Strategy

- construct a simulator $\mathcal{B}$ that signs the supplied messages in a way indistinguishable from the legitimate signer
- the attacker is able to perform forgery using only the simulator
- let us assume that $\mathcal{A}$ constructs a new verifiable tuple (M, R, S, T, U), M a new message, U not defined by the simulator
- use "forking lemma" to obtain two representations of $R = g^{E_G} Y^{E_Y} = g^{E'_G} Y^{E'_Y}$, which leads to X

Forking Lemma Details

- choose new random oracles and run the attack multiple times
- there is a constant probability that one find ℓ verifiable tuples $(M_i, R_i, S_i, T_i, U_i)$, such that U_i are pairwise distinct and
  - all R_i equal for TEGTSS-I
  - all (M_i, T_i) equal for TEGTSS-II

TEGTSS-I Simulator

- the simulator gets M
- computes $T = G(M)$
- chooses S and U at random
- computes R like in the verification algorithm
- defines $H(R) = U$ (this may fail)

TEGTSS-I Proof Sketch

- we have a verifiable tuple (M, R, S, T, U)
- case 1: $H(R)$ was defined by the simulator in the process of singing $M' \neq M$
  - the simulator result was (M', R, S', T', U')
  - $T = G(M) \neq G(M') = T'$ (collision resistance)
- case 2: $H(R)$ was defined by a direct query to H
  - by forking lemma, we have two tuples (M_1, R, S_1, T_1, U_1) and (M_2, R, S_2, T_2, U_2), U_i pairwise distinct
  - if F_3 equals for these tuples, we get $T_1 = G(M_1) = G(M_2) = T_2$, a contradiction
- by TEGTSS-I properties, $F_3 \neq F_3'$
- we get two representations of R

TEGTSS-I Proof Sketch

- we have a verifiable tuple (M, R, S, T, U)
- case 1: $H(R)$ was defined by the simulator in the process of singing $M' \neq M$
  - the simulator result was (M', R, S', T', U')
  - $T = G(M) \neq G(M') = T'$ (collision resistance)
- case 2: $H(R)$ was defined by a direct query to H
  - by forking lemma, we have two tuples (M_1, R, S_1, T_1, U_1) and (M_2, R, S_2, T_2, U_2), U_i pairwise distinct
  - if F_3 equals for these tuples, we get $T_1 = G(M_1) = G(M_2) = T_2$, a contradiction
- by TEGTSS-I properties, $F_3 \neq F_3'$
- we get two representations of R

TEGTSS-I Proof Sketch

- we have a verifiable tuple (M, R, S, T, U)
- case 1: $H(R)$ was defined by the simulator in the process of singing $M' \neq M$
  - the simulator result was (M', R, S', T', U')
  - $T = G(M) \neq G(M') = T'$ (collision resistance)
- case 2: $H(R)$ was defined by a direct query to H
  - by forking lemma, we have two tuples (M_1, R, S_1, T_1, U_1) and (M_2, R, S_2, T_2, U_2), U_i pairwise distinct
  - if F_3 equals for these tuples, we get $T_1 = G(M_1) = G(M_2) = T_2$, a contradiction
- by TEGTSS-I properties, $F_3 \neq F_3'$
- we get two representations of R

TEGTSS-I Proof Sketch

- we have a verifiable tuple (M, R, S, T, U)
- case 1: $H(R)$ was defined by the simulator in the process of singing $M' \neq M$
  - the simulator result was (M', R, S', T', U')
  - $T = G(M) \neq G(M') = T'$ (collision resistance)
- case 2: $H(R)$ was defined by a direct query to H
  - by forking lemma, we have two tuples (M_1, R, S_1, T_1, U_1) and (M_2, R, S_2, T_2, U_2), U_i pairwise distinct
  - if F_3 equals for these tuples, we get $T_1 = G(M_1) = G(M_2) = T_2$, a contradiction
- by TEGTSS-I properties, $F_3 \neq F_3'$
- we get two representations of R

Thanks

Thank you for your attention.