Stamp & Extend - Instant but Undeniable Timestamping based on Lazy Trees

Łukasz Krzywiecki, Przemysław Kubiak, Mirosław Kutyłowski
Wrocław University of Technology

InTrust 2012, London

According to the recent proposal for a regulation of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market:

"electronic time stamp" means data in electronic form which binds other electronic data to a particular time establishing *evidence* that these data existed at that time

Electronic time stamp

1. A digital signature provides guarantees for document origin, its aproval by the signatory, but it does not prove when the signature was created.

Electronic time stamp

1. A digital signature provides guarantees for document origin, its aproval by the signatory, but it does not prove when the signature was created.
2. Signing time is crucial for the legal consequences - e.g., in administrative procedures a party has a limited period of time to perform a legally valid action.

Electronic time stamp

1. A digital signature provides guarantees for document origin, its aproval by the signatory, but it does not prove when the signature was created.
2. Signing time is crucial for the legal consequences - e.g., in administrative procedures a party has a limited period of time to perform a legally valid action.
3. The recent proposal states that "Qualified electronic time stamp shall enjoy a *legal* presumption of ensuring the time it indicates and the integrity of the data to which the time is bound".

Trusted services

- A trusted service (TSA) uses a special purpose, secure time-stamping device.
- Technical security of the device, its resistance to manipulations is checked during certification process.

Trusted services

Krzywiecki, Kubiak, Kutyłowski

- A trusted service (TSA) uses a special purpose, secure time-stamping device.
- Technical security of the device, its resistance to manipulations is checked during certification process.

But:

- Certification process is only a process of checking of some properties against a certain list (a Protection Profile) that may ignore or overlook some important issues.
- TSA may itself be interested to retrieve the keys stored in the device to be able to backdate certain documents.

Honesty of TSA forced by the protocol

The basic structure - a linear chain of hashes

- Each element of the chain contains a signature of TSA on:

Honesty of TSA forced by the protocol

The basic structure - a linear chain of hashes

- Each element of the chain contains a signature of TSA on:
  - digital data to be stamped,

Honesty of TSA forced by the protocol

The basic structure - a linear chain of hashes

- Each element of the chain contains a signature of TSA on:
  - digital data to be stamped,
  - hash of the previous element in the chain.

Honesty of TSA forced by the protocol

The basic structure - a linear chain of hashes

- Each element of the chain contains a signature of TSA on:
  - digital data to be stamped,
  - hash of the previous element in the chain.
- The very first element of the chain is the certificate of TSA's public key.

Honesty of TSA forced by the protocol

The basic structure - a linear chain of hashes

- Each element of the chain contains a signature of TSA on:
  - digital data to be stamped,
  - hash of the previous element in the chain.
- The very first element of the chain is the certificate of TSA's public key.
- Disadvantage: verification time is linear in the number of time stamps issued.

Honesty of TSA forced by the protocol

Round schemes

- Time is split into rounds.

Honesty of TSA forced by the protocol

Round schemes

- Time is split into rounds.
- Within a round, TSA is executing a procedure that finally delivers a single value.

Honesty of TSA forced by the protocol

Round schemes

- Time is split into rounds.
- Within a round, TSA is executing a procedure that finally delivers a single value.
- The single value may be used in the next round to form a linear chain of rounds.

Honesty of TSA forced by the protocol

Round schemes

- Time is split into rounds.
- Within a round, TSA is executing a procedure that finally delivers a single value.
- The single value may be used in the next round to form a linear chain of rounds.
- Advantage: fast verification within a round.

Honesty of TSA forced by the protocol

Round schemes

- Time is split into rounds.
- Within a round, TSA is executing a procedure that finally delivers a single value.
- The single value may be used in the next round to form a linear chain of rounds.
- Advantage: fast verification within a round.
- Disadvantage: a requester of a timestamp must wait till the end of the round to obtain the proof that the timestamp is included in the final value of the round.

Honesty of TSA forced by the protocol

Round schemes

- Time is split into rounds.
- Within a round, TSA is executing a procedure that finally delivers a single value.
- The single value may be used in the next round to form a linear chain of rounds.
- Advantage: fast verification within a round.
- Disadvantage: a requester of a timestamp must wait till the end of the round to obtain the proof that the timestamp is included in the final value of the round.

Construction of a single round

one-way accumulators, aggregated signatures, Merkle trees.

Honesty of TSA forced by the protocol

Instant time-stamping

- Hashes of the requests are generated in advance - chameleon hash function h_c is used.

Honesty of TSA forced by the protocol

Instant time-stamping

- Hashes of the requests are generated in advance - chameleon hash function h_c is used.
- Merkle tree for the round is build before the first request is made.

Honesty of TSA forced by the protocol

Krzywiecki,
Kubiak,
Kutyłowski

Instant time-stamping

- Hashes of the requests are generated in advance - chameleon hash function h_c is used.
- Merkle tree for the round is build before the first request is made.
- The root of the tree is published.

Honesty of TSA forced by the protocol

Krzywiecki, Kubiak, Kutyłowski

Instant time-stamping

- Hashes of the requests are generated in advance - chameleon hash function h_c is used.
- Merkle tree for the round is build before the first request is made.
- The root of the tree is published.
- For each request m a value r is generated by the service in such a way $h_c(m, r)$ fits the first unused hash value generated in advance.

Honesty of TSA forced by the protocol

Instant time-stamping

- Hashes of the requests are generated in advance - chameleon hash function h_c is used.
- Merkle tree for the round is build before the first request is made.
- The root of the tree is published.
- For each request m a value r is generated by the service in such a way $h_c(m, r)$ fits the first unused hash value generated in advance.
- A trapdoor necessary to generate values r is distributed between a few servers. They must collude to backdate a document.

Honesty of TSA forced - our approach

Instant time-stamping - changes

- Instead of making commitments to the hashes of future requests we make commitments to randomness used in signatures under answers to the requests.

Honesty of TSA forced - our approach

Instant time-stamping - changes

- Instead of making commitments to the hashes of future requests we make commitments to randomness used in signatures under answers to the requests.
- Tree of commitments is made gradually, when consecutive requests are answered (unlimited size of the tree).

Honesty of TSA forced - our approach

Instant time-stamping - changes

- Instead of making commitments to the hashes of future requests we make commitments to randomness used in signatures under answers to the requests.
- Tree of commitments is made gradually, when consecutive requests are answered (unlimited size of the tree).
- **If the same randomness is used to sign answers to two different requests then the private key of TSA leaks.**

Honesty of TSA forced - our approach

Instant time-stamping - changes

- Instead of making commitments to the hashes of future requests we make commitments to randomness used in signatures under answers to the requests.
- Tree of commitments is made gradually, when consecutive requests are answered (unlimited size of the tree).
- **If the same randomness is used to sign answers to two different requests then the private key of TSA leaks.**
- Accordingly, we have an undeniable evidence that: private key of TSA is used outside the TSA, or TSA is misbehaving.

Honesty of TSA forced - our approach

Consequences

- TSA is dettered from misbehaviour (TSA is centralized).

Honesty of TSA forced - our approach

Krzywiecki, Kubiak, Kutyłowski

Consequences

- TSA is dettered from misbehaviour (TSA is centralized).
- Costly certification process of the time-stamping device is not necessary - the protocol provides evidence of a fraud.

Honesty of TSA forced - our approach

Consequences

- TSA is dettered from misbehaviour (TSA is centralized).
- Costly certification process of the time-stamping device is not necessary - the protocol provides evidence of a fraud.
- Each request is served instantly.

Honesty of TSA forced - our approach

Consequences

- TSA is dettered from misbehaviour (TSA is centralized).
- Costly certification process of the time-stamping device is not necessary - the protocol provides evidence of a fraud.
- Each request is served instantly.
- Any two timestamps are comparable with respect to the order they were requested.

Protocol's Building Blocks - Schnorr Signatures

Keys

Private key: x, public key: g^x, where $\langle g \rangle$ is a group of prime order q, in which DLP is hard.

Protocol's Building Blocks - Schnorr Signatures

Keys

Private key: x, public key: g^x, where $\langle g \rangle$ is a group of prime order q, in which DLP is hard.

Signature generation

1. the signer chooses an integer $k \in [1, q-1]$ uniformly at random,
2. $r := g^k$,
3. $e := H(M||r)$ ($||$ stands for concatenation),
4. $s := (k - xe) \bmod q$,
5. output signature (e, s).

Protocol's Building Blocks - Schnorr Signatures

Keys

Private key: x, public key: g^x, where $\langle g \rangle$ is a group of prime order q, in which DLP is hard.

Signature generation

1. the signer chooses an integer $k \in [1, q-1]$ uniformly at random,
2. $r := g^k$,
3. $e := H(M||r)$ ($||$ stands for concatenation),
4. $s := (k - xe) \bmod q$,
5. output signature (e, s).

Note: **if the same** k **is used twice, for different** M, M', **then key** x **leaks!**

Protocol's Building Blocks - Pedersen commitments

Assumption

Let $h \in \langle g \rangle$ such that $\log_g h$ is known to nobody.

Protocol's Building Blocks - Pedersen commitments

Krzywiecki, Kubiak, Kutyłowski

Assumption

Let $h \in \langle g \rangle$ such that $\log_g h$ is known to nobody.

Commitment

- Commitment c to k is obtained by choosing $\ell \in \{0, 1, \ldots, q-1\}$ uniformly at random and assigning:

$$c := g^k \cdot h^\ell.$$

- Commitment c reveals no information about k.
- Changing the commitment c to a k' such that $k' \neq k$ implies knowledge of $\log_g h$. Therefore it is infeasible to replace k by k'.

The protocol

Certificate HS_0 of TSA contains y, and c_1 where:

- $y = g^x$ is TSA's public, signature verification key,
- $c_1 = g^{k_1} h^{\ell_1}$ is the first commitment, where k_1, ℓ_1 are uniformly chosen.

The protocol

Certificate HS_0 of TSA contains y, and c_1 where:

- $y = g^x$ is TSA's public, signature verification key,
- $c_1 = g^{k_1} h^{\ell_1}$ is the first commitment, where k_1, ℓ_1 are uniformly chosen.

Data stored by TSA

- the index of the last timestamp issued $i - 1$ (initially $i = 1$),
- a private list P of pairs of exponents $[(k_i, \ell_i), \ldots, (k_{2i-1}, \ell_{2i-1})]$
- a public file C with the list of Pedersen commitments $[c_1, \ldots, c_{2i-1}]$,
- a public file HS that includes an initial value HS_0 and timestamps HS_j for $j = 1, \ldots, i - 1$.

The protocol - processing a request H_i by TSA

1. choose $k_{2i}, \ell_{2i}, k_{2i+1}, \ell_{2i+1}$ uniformly at random

The protocol - processing a request H_i by TSA

Krzywiecki, Kubiak, Kutyłowski

1. choose $k_{2i}, \ell_{2i}, k_{2i+1}, \ell_{2i+1}$ uniformly at random
2. $c_{2i} := g^{k_{2i}} h^{\ell_{2i}}, \quad c_{2i+1} := g^{k_{2i+1}} h^{\ell_{2i+1}}$

The protocol - processing a request H_i by TSA

1. choose $k_{2i}, \ell_{2i}, k_{2i+1}, \ell_{2i+1}$ uniformly at random
2. $c_{2i} := g^{k_{2i}} h^{\ell_{2i}}, \quad c_{2i+1} := g^{k_{2i+1}} h^{\ell_{2i+1}}$
3. append c_{2i}, c_{2i+1} to C
4. $k := k_i$, remove (k_i, ℓ_i) from P, append $(k_{2i}, \ell_{2i}), (k_{2i+1}, \ell_{2i+1})$ to P

The protocol - processing a request H_i by TSA

1. choose $k_{2i}, \ell_{2i}, k_{2i+1}, \ell_{2i+1}$ uniformly at random
2. $c_{2i} := g^{k_{2i}} h^{\ell_{2i}}, \quad c_{2i+1} := g^{k_{2i+1}} h^{\ell_{2i+1}}$
3. append c_{2i}, c_{2i+1} to C
4. $k := k_i$, remove (k_i, ℓ_i) from P, append $(k_{2i}, \ell_{2i}), (k_{2i+1}, \ell_{2i+1})$ to P
5. using k create Schnorr signature (e_i, s_i) on "message":

$$(H(\mathrm{HS}_{i-1}), \mathrm{H}_i, c_{2i}, c_{2i+1}, \ell_i, i)$$

The protocol - processing a request H_i by TSA

1. choose $k_{2i}, \ell_{2i}, k_{2i+1}, \ell_{2i+1}$ uniformly at random
2. $c_{2i} := g^{k_{2i}} h^{\ell_{2i}}, \quad c_{2i+1} := g^{k_{2i+1}} h^{\ell_{2i+1}}$
3. append c_{2i}, c_{2i+1} to C
4. $k := k_i$, remove (k_i, ℓ_i) from P, append $(k_{2i}, \ell_{2i}), (k_{2i+1}, \ell_{2i+1})$ to P
5. using k create Schnorr signature (e_i, s_i) on "message":

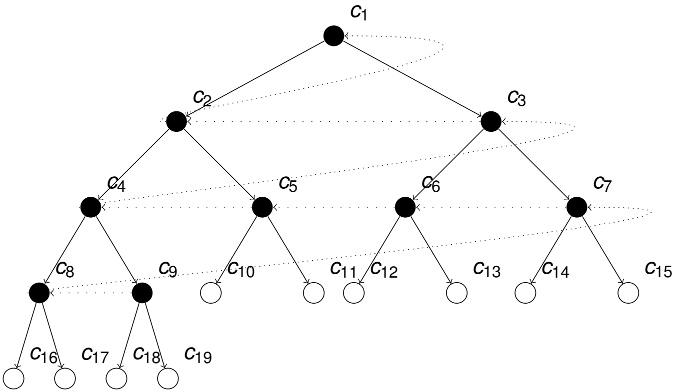
$$(H(\mathrm{HS}_{i-1}), \mathrm{H}_i, c_{2i}, c_{2i+1}, \ell_i, i)$$

6. return the sequence of records to the requester

$$((e_i, s_i), H(\mathrm{HS}_{j-1}), \mathrm{H}_j, c_{2j}, c_{2j+1}, \ell_j, j) \tag{1}$$

for $j = \lfloor i/2^\alpha \rfloor$, where $\alpha = 0, 1, \ldots, \lfloor \log_2 i \rfloor$.

Two structures fused, $i = 9$



The protocol: the main trick ...

- If the same commitment c_i is utilized twice for signing two different requests H_i, H_i' then the private key leaks (see the second component of Schnorr signatures).
- "An escape route" for the forger would be to change commitments, but then . . .

The protocol: ... the main trick ...

- Assign $c'_j := g^{e_j} y^{s_j} h^{\ell_j}$ for $j = \lfloor i/2^\alpha \rfloor$, where $\alpha = 0, 1, \ldots, \lfloor \log_2 i \rfloor$ - see records (1).
- Note that if the sequence

$$c'_i, c'_{\lfloor i/2 \rfloor}, \ldots, c'_{\lfloor i/2^{\lfloor \log_2 i \rfloor - 2} \rfloor}, c'_{\lfloor i/2^{\lfloor \log_2 i \rfloor - 1} \rfloor}, c_1$$

is different from the publicly available sequence

$$c_i, c_{\lfloor i/2 \rfloor}, \ldots, c_{\lfloor i/2^{\lfloor \log_2 i \rfloor - 2} \rfloor}, c_{\lfloor i/2^{\lfloor \log_2 i \rfloor - 1} \rfloor}, c_1$$

then there is some index for which the sequences differ. By β denote the first such index counting from the right.
- Then $c_\beta \neq c'_\beta$, but $c_{\lfloor \beta/2 \rfloor} = c'_{\lfloor \beta/2 \rfloor}$ (at worst $\lfloor \beta/2 \rfloor = 1$).

The protocol: ...the main trick

- Hence the corresponding "messages" for $i = \lfloor \beta/2 \rfloor$ are different, because $c_\beta \neq c'_\beta$.

The protocol: ...the main trick

- Hence the corresponding "messages" for $i = \lfloor \beta/2 \rfloor$ are different, because $c_\beta \neq c'_\beta$.
- But the randomness used to make the signatures under the "messages" is the same, because $c_{\lfloor \beta/2 \rfloor} = c'_{\lfloor \beta/2 \rfloor}$.

The protocol: ...the main trick

- Hence the corresponding "messages" for $i = \lfloor \beta/2 \rfloor$ are different, because $c_\beta \neq c'_\beta$.
- But the randomness used to make the signatures under the "messages" is the same, because $c_{\lfloor \beta/2 \rfloor} = c'_{\lfloor \beta/2 \rfloor}$.
- Assuming that Schnorr signatures are hard to repudiate this leads to leakage of key x.

The protocol: requester's actions

- Each requester receiving a timestamp (i.e., each client application) should always verify *a constant* number n_{ver} of timestamps: the one received and $n_{ver} - 1$ consecutive predecessors of a randomly chosen timestamp in the chain (the random choice is made by the requester).

The protocol: requester's actions

- Each requester receiving a timestamp (i.e., each client application) should always verify *a constant* number n_{ver} of timestamps: the one received and $n_{ver} - 1$ consecutive predecessors of a randomly chosen timestamp in the chain (the random choice is made by the requester).
- We may assume that a local copy of all timestamps received is maintained by the requester, and a locally stored timestamp is compared with the newly received one if both are on the same position in the hash chain.

Thanks for your attention!

This work has been partially supported by Foundation for Polish Science - MISTRZ project.

FNP

Fundacja na rzecz Nauki Polskiej