Protecting Electronic Signatures in Case of Key Leakage

Mirosław Kutyłowski, Jacek Cichoń, Lucjan Hanzlik, Kamil Kluczniak,
Xiaofeng Chen, Jianfeng Wang

Wrocław University of Science and Technology, Poland
Xidian University, P.R.C.

MYCRYPT -Paradigm Shifting Cryptography 2016, Kuala Lumpur

Undeniability of electronic signatures

ideal world

Undeniability based on the assumptions:

1. creating a valid signature **only with the secret key** corresponding to the public key used during verification
2. the private key implemented in *signature creation device* **only**
3. the device under a **sole control** of the signatory,
4. the **link** between the verification key and the signatory is established

creating a valid signature only with the secret key corresponding to the public key used during verification

Reality

- strong research
- formal proofs – provable security
- reduction to cryptographic assumptions,

creating a valid signature only with the secret key corresponding to the public key used during verification

Reality

- strong research
- formal proofs – provable security
- reduction to cryptographic assumptions,

but

- what is the state-of-the-art? (not the public one)
- how can an end-user believe the cryptographers? so finally: **it is based on trust...**

Undeniability of electronic signatures

real world

the private key implemented in signature creation device only

Reality

- tamper resistance is hard to achieve ...
- ... but **even harder to provide an evidence** about it
- what about trapdoors, subliminal channels, etc. ?

Undeniability of electronic signatures

real world

the device under a sole control of the signatory

PIN/biometry/...

- so far PIN protection
- security level ...
- a quite secure solution based on mediated signatures, **but not deployed in practice ...**

Undeniability of electronic signatures

real world

the link between the verification key and the signatory is established

PKI

- theoretically works, but
- ... an Achilles Heel in practice
- again based on **unconditional trust**:
  what if a rogue CA generates a key pair and issues a certificate with the victim's name?

Generation of signing keys

Options

key generated by ...

1. **... a service provider and installed on the signing device:**
   rogue SP retains the keys and forges signatures
   (retaining keys might be even legal – EIDAS)
2. **... the user and installed on the signature creation device:**
   forbidden by law: opportunities to steal the key by rogue software and/or misbehavior of the user
3. **... the signature creation device:**
   does it really generate itself? Or it uses a pre-installed/kleptographic/weak key?

Key security

Conclusion

- **no real guarantees that the original signing keys are not in hand of rogue third parties**
- what can we do about it? Is it hopeless?

Key security

Conclusion

- **no real guarantees that the original signing keys are not in hand of rogue third parties**
- what can we do about it? Is it hopeless?

Our goal

build SOME countermeasures that might work in practice

Fail-stop signatures

- prevent **cryptanalytic attacks**
- **useless** against an adversary that holds the **original signing keys**

Mediated signatures/key evolution

- creation of a signature requires at least **2 devices**
- one of them could be a server implementing **additional security layer**
  analogous to monitoring activity of the credit cards
- evolution/fluctuation of keys on both sides to detect/disable **clones**

... still **limited practical deployment** despite tremendous progress in telecommunication

Smart cards for client-bank communication
application case

Crucial functionalities

- authentication of the client
- signing transactions for evidence purposes

Smart cards for client-bank communication
application case

Crucial functionalities

- authentication of the client
- signing transactions for evidence purposes

Problems

- **the bank should issue the card**, as it knows the customer
- **the bank should not issue the card** as in this case e-signatures have a limited value in a court of law – a third party should be involved

It is hard to make a change

problems to make any radical change

- high number of embedded devices that cannot be updates to new solutions
- tons of software/protocols based on previous solutions
- standards
- existing certificates

... the e-signatures do not work in practice for signing documents but a lot of resistance to make any change

Overall architecture

General assumptions

1. no changes in **standards** for electronic signatures

Overall architecture

General assumptions

1. no changes in **standards** for electronic signatures
2. no changes in (regular) **verification** procedures

Overall architecture

General assumptions

1. no changes in **standards** for electronic signatures
2. no changes in (regular) **verification** procedures
3. effective even against **manipulated PRNG** on the smart card

Overall architecture

General assumptions

1. no changes in **standards** for electronic signatures
2. no changes in (regular) **verification** procedures
3. effective even against **manipulated PRNG** on the smart card
4. effective even if the provider of the cards **retains** the signing keys

Overall architecture

General assumptions

1. no changes in **standards** for electronic signatures
2. no changes in (regular) **verification** procedures
3. effective even against **manipulated PRNG** on the smart card
4. effective even if the provider of the cards **retains** the signing keys
5. simple enough to be **understood by an average** IT engineer

Overall architecture

General assumptions

1. no changes in **standards** for electronic signatures
2. no changes in (regular) **verification** procedures
3. effective even against **manipulated PRNG** on the smart card
4. effective even if the provider of the cards **retains** the signing keys
5. simple enough to be **understood by an average** IT engineer
6. forgery **with the original keys** detectable with a pbb high enough to discourage the attacker

Application model

card life cycle

- service provider delivers the cards, private key generated as usual
- the user installs **hidden key**
- regular use:
  the device returns a signature created **according to the hidden key**
- the user detects a forged signature with his name:
  1. forgery **detection**
  2. **proving** forgery in front of a **judge**

Solution scheme

Signatures concerned

- based on Discrete Logarithm Problem,
- ... where the first step is to compute $r := g^k$ for a random k
- ... and where r is either a **part of the signature** or can be **reconstructed** by the verifier

Generating a key pair for a user

Almost no change:

- `signing device` stores a private signing key $x < q$ chosen at random,
- the public key $Y = g^x$ has been exported outside `signing device`,
- `signing device` is in the state requiring installing the hidden control keys.

Installing the hidden control keys

Executed by the user interacting with his `signing device` with already instantiated private signing key x.

1. the user
   - chooses the hidden secret key $v < q$, at random,
   - computes $V := g^v$
2. the user **authenticates** himself against the device `signing device` (PIN etc) and **uploads** V to `signing device`
3. `signing device` **ready** for creating signatures.
4. the user **creates a few signatures** and **deposits** them in a trusted place

Signing procedure

Creation of Schnorr signatures

1. choose k at random
2. $r := g^k$
3. $e := \mathrm{Hash}(M||r)$
4. $s := (k + x \cdot e) \bmod q$
5. output (s, e) as a signature of M.

Signing procedure
modified

Phase 1 (preprocessing) :

1. create an empty array $A[0 \ldots 59]$
2. choose k at random
3. $U := V^k$
4. $i := 0$
5. repeat Δ times:

$$z := \mathrm{TruncHash}(U, M)$$
$$A[z] := i$$
$$i := i + 1$$
$$U := U \cdot V$$

array A and k retained

Phase 2 (the signing part) :

1. T = the UTC signing time, t = seconds
2. wait until $A[t]$ is nonempty
3. $r := g^{k+A[t]}$
4. having r already computed proceed as before

Forgery detection

test

1. reconstruct r,
   e.g. $r := g^s / Y^e$ for a Schnorr signature (s, e)
2. check

$$\mathrm{TruncHash}(r^v, M) \stackrel{?}{=} t$$

where colorred $t =$ seconds of the signing time

- secret hidden key v needed for the test
- based on equality $r^v = (g^k)^v = (g^v)^k = V^k$

Forgery proof

1. r reconstructed
2. the user computes $u := r^v$ and presents to the judge.
3. the forgery claim rejected if $\mathrm{TruncHash}(u, M) = t$
4. the user and the judge perform an interactive ZKP of equality of discrete logs for (g, V) and (r, u). E.g.:
   1. the user chooses σ at random and presents
   $$v_1 = g^{v\sigma}, v_2 = r^{v\sigma}$$
   2. the judge chooses a bit b at random,
   3. if $b = 0$, then the user reveals σ and the judge checks that $v_1 = V^{\sigma}, v_2 = u^{\sigma}$,
   4. if $b = 1$, then the user reveals $\delta = v\sigma$ and the judge checks that $v_1 = g^{\delta}, v_2 = r^{\delta}$.
5. if ZKP succeeds, then the judge recognizes forgery

Security

adversaries

- signatory
- device
- manufacturer
- verifier

threats

- modified procedure may simplify forgery
- hidden key may be reconstructed
- device may leak the hidden key V

Resilience to forgeries

reduction argument

breaking the original scheme if the proposed one broken:

- choose v and $V = g^v$
- run the device, delete all signatures where forgery would be detected
- feed the remaining ones as input to the forgery procedure
- receive its output - a forged signature

Indistinguishability

Theorem

An observer cannot decide whether he gets signatures from the original scheme or from the proposed one.

Assumption - negligible advantage in Correlated TruncHash Values Game

1. choose pairwise different elements $k_1, \ldots, k_n \leq q$
2. choose $M_1, \ldots, M_n \in \mathcal{G}$
3. choose V at random
4. $h_i := \mathrm{TruncHash}(V^{k_i}, M_i)$ for $i = 1$ to n,
5. choose M and $k \neq k_1, \ldots, k_n$
6. $h^{(0)}_{n+1} := \mathrm{TruncHash}(V^k, M)$
7. choose $h^{(1)}_{n+1} \in \{0, \ldots, 59\} \setminus \{h^{(0)}_{n+1}\}$ at random
8. choose $b \in \{0, 1\}$ at random
9. $\hat{b} := \mathcal{A}(k_1, \ldots, k_n, k, M_1, \ldots, M_n, M, h_1, \ldots, h_n, h^{(b)}_{n+1})$.

$\mathcal{A}$ wins the game, if $b = \hat{b}$.

The dark side of the scheme

attack

the signing device may implement a similar approach to leak the secret key if installed there by a third trusted party:

- V^k used to determine the position of the bit leaked
- the last bit of r should be equal to the key-bit on this position
- if this is not true than the next r generated and the signature created

chances $\frac{3}{4}$ that r indicates the key-bit correctly

The attacker observes the signatures and creates statistics for each key-bit position.

The dark side of the scheme

Corollary

- the PRNG might be honest, perfect, separated in hardware (no room for a kleptographic channel)
- the keys might be created honestly (e.g. cliptographic method)
- but nevertheless timing may be used to create a subliminal channel by subverted software on the signing device

Recommendation

we better make the signing time less precise

Thanks for your attention!

Contact data

1. Miroslaw.Kutylowski@pwr.edu.pl
2. http://kutylowski.im.pwr.edu.pl
3. http://cs.pwr.edu.pl