Strengthened Security for Blind Signatures

David Pointcheval

Laboratoire d'Informatique
École Normale Suprieure

David.Pointcheval@ens.fr
http://www.dmi.ens.fr/~pointche

Summary

Blind Signatures

An authority helps a user to get a valid signature

the message and the signature
must remain unknown for the authority

$\Longrightarrow$ (revokable) anonymity

- e–cash
- e–voting

Security Properties

- $(\ell, \ell+1)$-**forgery:** after ℓ interactions with the authority the attacker can forge $\ell+1$ message–signature valid pairs.

Attacks

- **Sequential attack:** the attacker interacts sequentially with the signer.
- **Parallel attack:** the attacker can initiate several interactions at the same time with the signer, in any order he wants.

Previous Results

- **Complexity-Based Security:** [Da-89], [PfWa-91] and recently [JuLuOs-97] proved the existence of secure schemes using secure signature schemes and multi-party computation

  $\Longrightarrow$ totally inefficient and impractical

- **Random Oracle Model:** [PS-96] proposed the first arguments towards secure and efficient schemes using witness-indistinguishability
  (WI is required for the simulation of the signer).

Okamoto–Schnorr Blind Scheme

The signer – Σ		Alice
Common: p, q, g, h Keys: $y = g^{-r} h^{-s} \bmod p$		Message to sign: m
$t, u \in \mathbb{Z}_q^\star$ $a = g^t h^u \bmod p$	$\xrightarrow{\quad a \quad}$	$\beta, \gamma, \delta \in \mathbb{Z}_q$ $\alpha = a g^\beta h^\gamma y^\delta \bmod p$ $\varepsilon = H(m, \alpha)$
	$\xleftarrow{\quad e \quad}$	$e = \varepsilon - \delta \bmod q$
$R = t + er \bmod q$ $S = u + es \bmod q$	$\xrightarrow{\quad R, S \quad}$	$g^R h^S y^e \stackrel{?}{=} a \bmod p$ $\rho = R + \beta \bmod q$ $\sigma = S + \gamma \bmod q$

$(m, \alpha, \varepsilon, \rho, \sigma)$ s.t. $\alpha = g^\rho h^\sigma y^\varepsilon \bmod p$ with $\varepsilon = H(m, \alpha)$.

Previous Result

If $\mathcal{A}$ is a Turing Machine which can perform an $(\ell, \ell+1)$-forgery, under a parallel attack,

- after Q queries to the random oracle h,
- after R initiated interactions with the signer, (but only ℓ completed ones),
- with probability $\varepsilon \geq 4Q^{\ell+1}R^{\ell}/q$.

The Discrete Logarithm Problem can be solved

- after $33Q\ell/\varepsilon$ calls to $\mathcal{A}$
- with probability greater than $\frac{1}{72\ell^2}$.

Asymptotically

Let k be the security parameter.

Let us assume that $|q| = k$.

If $\ell \ll k/\log k$, for any polynomials P, Q and A,

$4Q^{\ell+1}R^{\ell}/q \leq 1/A$, for k large enough.

$\Longrightarrow$ If $\mathcal{A}$ works within polynomial time T, with non-negligible probability of success ε, then for any ℓ poly-logarithmically bounded, the Discrete Logarithm Problem can be solved within time $2376\ell^3 T/\varepsilon$, for any k large enough.

Generic Transformation

It is a kind of "cut-and-choose":

- one duplicates everything except the final answer
- one asks the user to commit its "blinding" factors
- after the 2 queries:
  the authority randomly chooses one, $I \in_R \{0,1\}$
  and checks its well-done construction
  then answers the other query, e_{1-I}.

The signer		Alice
Common: p, q, g, h Keys: $y = g^{-r}h^{-s} \bmod p$		$i = 0, 1$ and $J \stackrel{\text{def}}{=} 1 - I$ $\beta_i, \gamma_i, \delta_i \in \mathbb{Z}_q$ ϕ_i, ψ_i random, $\mu_i = H(m, \phi_i)$ $h_i = H(\beta_i, \gamma_i, \delta_i, \mu_i, \psi_i)$
$t_i, u_i \in \mathbb{Z}_q$ $a_i = g^{t_i}h^{u_i} \bmod p$	$\xleftarrow{h_i}$ $\xrightarrow{a_i}$	$\alpha_i = a_i g^{\beta_i} h^{\gamma_i} y^{\delta_i} \bmod p$ $e_i = H(\mu_i, \alpha_i) - \delta_i \bmod q$
$I \in \{0,1\}$	$\xleftarrow{e_i}$ $\xrightarrow{I}$	
Verification of h_I and e_I $R = t_J + e_J \cdot r \bmod q$ $S = u_J + e_J \cdot s \bmod q$	$\xleftarrow{\beta_I, \gamma_I, \delta_I, \mu_I, \psi_I}$ $\xrightarrow{R, S}$	$a_J \stackrel{?}{=} g^R h^S y^{e_J} \bmod p$ $\rho = R + \beta_J \bmod q$ $\sigma = S + \gamma_J \bmod q$

Then $\alpha = g^\rho h^\sigma y^\varepsilon \bmod p$, $\mu = H(m, \phi)$ and $\varepsilon = H(\mu, \alpha)$
where $\alpha = \alpha_J$ and $\phi = \phi_J$

Claim

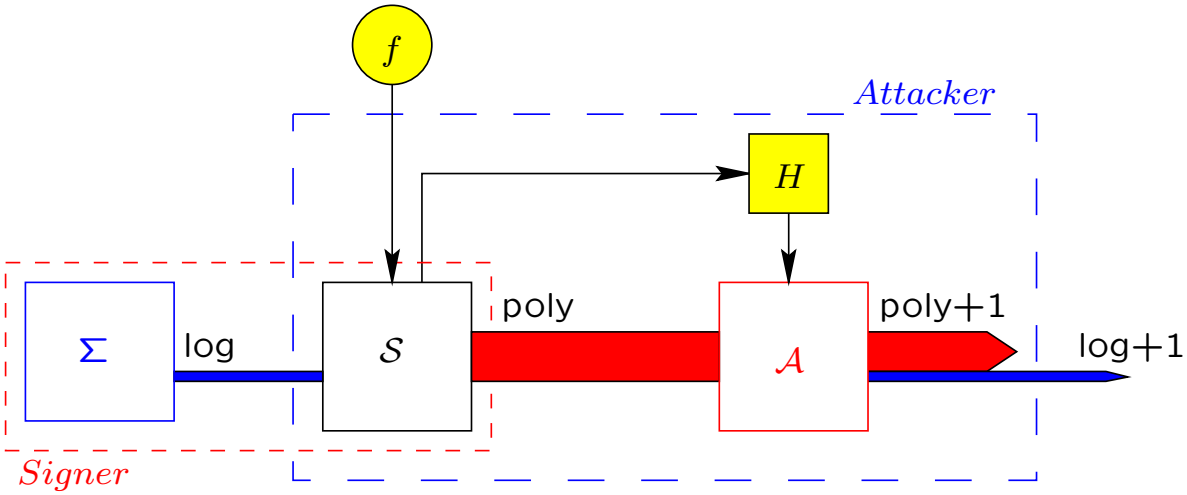
- **Synchronized Parallel Attack:** the attacker can initiate several interactions at the same time with the signer, but for each round, indexes follow the same order.

  seq. attack $<$ synchr. parallel attack $<$ parallel attack

- **Security:** If there exist polynomials ℓ, Q and P, and a Turing Machine $\mathcal{A}$ which can perform
  an $(\ell, \ell+1)$-forgery,
  under a synchronized parallel attack,
  - after Q queries to the random oracle h,
  - with probability $\varepsilon \geq 1/\mathbf{P}$.

  The Discrete Logarithm Problem can be solved
  - after $\mathcal{O}(\log k)Q/\varepsilon$ calls to $\mathcal{A}$
  - with probability greater than $\Omega(1/(\log k)^2)$.

Reduction



- New scheme
- OS scheme

$Signer$	signer	
$\mathcal{A}$	attacker	
Σ	signer	
$Attacker$	attacker	

- $\mathcal{S}$ Simulator
- f random oracle
- H $\mathcal{S}$-controled random oracle

The Simulator $\mathcal{S}$

- $\mathcal{S}$ randomly chooses $j \in \{0,1\}$:
  1. $\mathcal{S}$ performs a stand-alone simulation for $i = 1 - j$:
     randomly choosing the challenge $w \implies a_{1-j}$
     looking in the table of f, define $H(\mu_i, \alpha_i)$ to be asked for w
  2. $\mathcal{S}$ asks for some help to Σ for $i = j \implies a_j$

  $\mathcal{S}$ sends a_0 and a_1 to $\mathcal{A}$
- $\mathcal{A}$ sends the challenges e_0 and e_1
- $\mathcal{S}$ can check with the expected challenges
  (looking at the queries to f)
  If the attacker has played honestly then $\mathcal{S}$ defines $I = j$,
  else it lets $I = 1 - j$, and asks I
- $\mathcal{A}$ reveals the blinding factors
- $\mathcal{S}$ checks the commitment `False`: $\mathcal{S}$ stops the game
  `True`: if $I = j$
  then $\mathcal{S}$ ends its simulation
  else $\mathcal{S}$ sends $\Sigma(e_{1-I}) = (R, S)$.

Properties

Let us assume that $\mathcal{A}$ can perform an $(\ell, \ell+1)$-forgery
against $Signer$ under a **synchronized parallel attack**
for ℓ **polynomially bounded**.

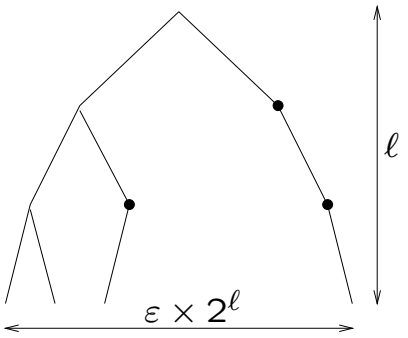
The number of initiated interactions with Σ is equal to ℓ.
We denote by λ the number of completed interactions with Σ.

1. $\mathcal{A}$ cannot distinguish $\Sigma \cup \mathcal{S}$ from $Signer$:
   the challenge "I" is equal to $j \oplus v$,
   where $j \in_R \{0,1\}$ and $v =$ "has $\mathcal{A}$ played honestly?"
   (and v independent of j).

2. The number of valid signatures (w.r.t. f) is greater than $\lambda+1$:

$\varepsilon = H(\mu,\alpha) \neq f(\mu,\alpha) \implies \varepsilon = H(\mu,\alpha)$ defined by $\mathcal{S}$
$\implies \mathcal{S}$ has simulated everything $\implies$ no help from Σ
$\#\{\text{valid signatures}\} = \ell + 1 - \#\{\varepsilon \neq f(\mu,\alpha)\}$
$\geq \ell + 1 - (\ell - \lambda) \geq \lambda + 1$

3. With constant probability, λ is logarithmically bounded:



• = single node

Help of $\Sigma \implies \mathcal{A}$ has not played honestly
$\implies$ single node (or collision for f).

So Pr[less than $\log(2/\varepsilon)$ • | leaf] $\geq 1/2$

Consequences

- Assumption: $\mathcal{A}$ can perform an $(\ell, \ell+1)$-forgery
  against *Signer* under a synchronized parallel attack
  (Q queries, probability ε).
- Consequence: $\mathcal{S} \cup \mathcal{A}$ can perform an $(\lambda, \lambda+1)$-forgery
  against Σ under a parallel attack
  (Q queries, probability $\varepsilon' \geq \varepsilon/16$)
  after ℓ initiated interactions but $\lambda \leq \log(4/\varepsilon)$ completed ones

If ε is non-negligible, and Q, ℓ polynomially bounded,
for any k large enough, $\varepsilon' \geq \varepsilon/16 \geq 4Q^{\lambda+1}\ell^{\lambda}/q$

Then the Discrete Logarithm Problem can be solved
- with probability greater then $\Omega(1/(\log k)^2)$
- after less than $\mathcal{O}(\log k)Q/\varepsilon$ steps.

Conclusion

With a kind of "cut-and-choose",
we impose the user to play honestly.

A dishonest user will be detected
before it is too late.

We have presented a generic transformation which

- makes secure:
  after polynomially many synchronized interactions
  against poly-logarithmically many attackers.
- remains practical and efficient.
  the output signature is an OS signature

This transformation can be adapted
to many other WI-based blind signature schemes