Strong and provable secure ElGamal type signatures

Chapter 16.3

Santeri.Saarinen@hut.fi

Overview

- El Gamal cryptosystem and El Gamal type signatures
- Terms used
- Forking reduction
- Discussion on the results
- Heavy-Row reduction
- Conclusion

El Gamal cryptosystem

- Public key system based on discrete logarithm problem
- Prime p and primitive element α
- Private key is a and $\beta = \alpha^{a} \bmod p$
- Random number k, message x
- E: $y_1 = \alpha^{k} \bmod p$, $y_2 = x*\beta^{k} \bmod p$
- D: $y_2 * (y_1^{a})^{-1} \bmod p$

El Gamal example - encryption

- Suppose: $p = 13$, $\alpha = 2$, $a = 3$,
  $\beta = 2^3 \bmod 13 = 8$,
  message $x = 11$, random $k = 5$
- Public key: $\{p, \alpha, \beta\} = \{13, 2, 8\}$
- Encryption: $y_1 = 2^5 \bmod 13 = 6$
  $y2 = 11 * 8^5 \bmod 13 = 10$
- Ciphertext: (6, 10)

El Gamal example - decryption

- Public key: $\{p, \alpha, \beta\} = \{13, 2, 8\}$
- Private key: $a = 3$
- Ciphertext: $y = \{6, 10\}$
- $x = 10 * (6^3)^{-1} \bmod 13 = 11$

El Gamal signature scheme

- sig(m, k) = (y_1, y_2)
  $y_1 = \alpha^{k} \bmod p$
  $y_2 = (m - a\, y_1)(k^{-1}) \bmod (p - 1)$
- ver(m, y_1, y_2) $\Leftrightarrow$
  $y_1^{y2} * \beta^{y1} = \alpha^{m} \bmod p$

El Gamal signature example

- Public key: $\{p, \alpha, \beta\} = \{13, 2, 8\}$
- Private key: $a = 3$
- $m = 11, k = 5$
- sig(11, 5):
  $y_1 = \alpha^{k} \bmod p = 6$
  $y_2 = (m - a\, y_1)(k^{-1}) \bmod (p - 1) = 1$

El Gamal Signature example cont.

- verify:
  $y_1^{y2} * \beta^{y1} = \alpha^{m} \bmod p$
  $6^1 * 8^6 = 2^{11} \bmod 13$
  $7 = 7 \Leftrightarrow$ true
- Digital Signature Algorithm (DSA) and Schnorr are variants of El Gamal.

Triplet Signature Scheme

- Signature of message M is triplet (r,e,s)
- r is called *commitment*, committing epheremal integer l. Constructed for example: $r = g^l \bmod p$
- e = H(M, r), where H() is a hash function
- s is called signature, a linear function of (r, l, M, H(), signing key)

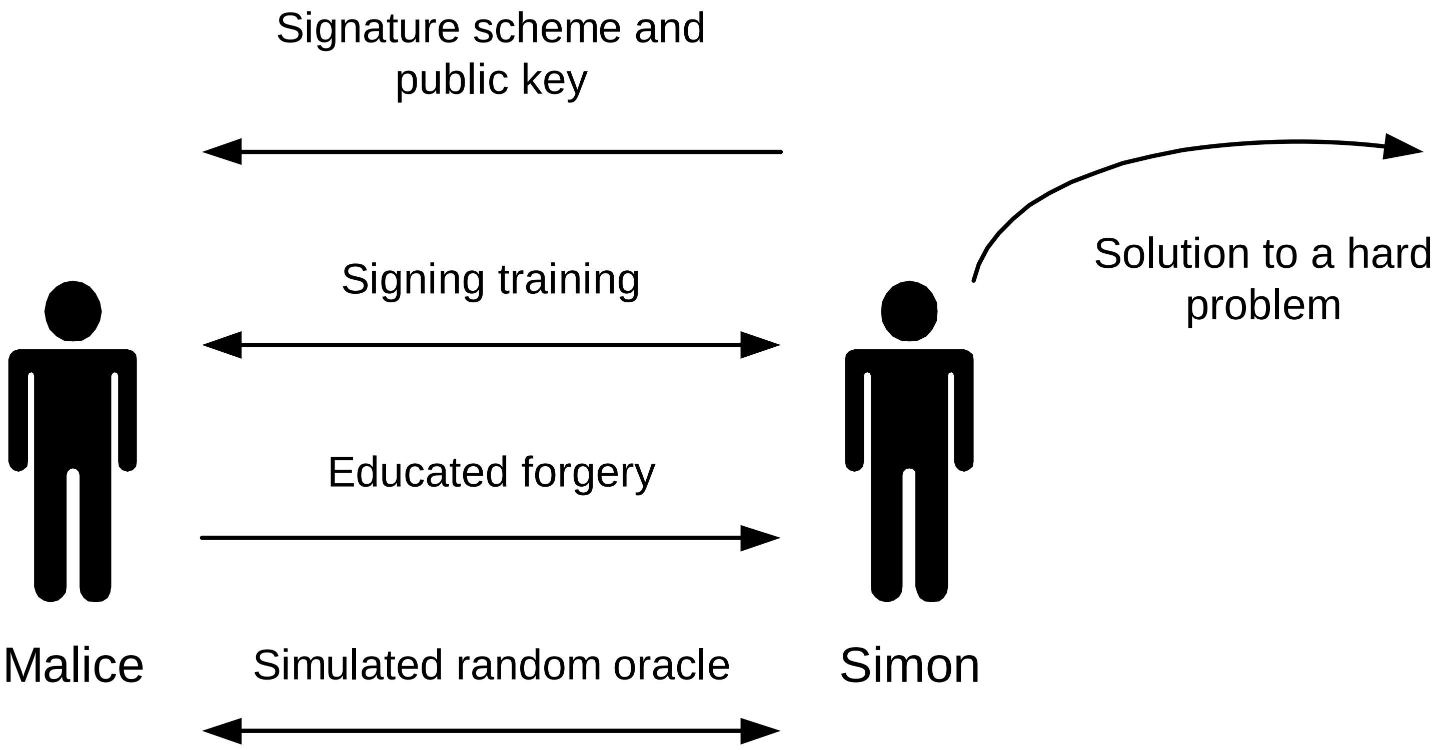
Secure Signature Scheme

- Signature scheme is denoted by (*Gen*, *Sign*, *Verify*)
- *Gen* generates private and public key
- *Sign* signs message and *Verify* verifies
- Signature scheme is ($t(k)$, $Adv(k)$), if there exists no forger able to forge a signature for all sufficiently large k.

Reduction

- Transformation from $t(k)$, $Adv(k)$ to $t'(k)$, $Adv'(k)$, which is corresponding solution to a hard problem (e.g. discrete logarithm)
- Main aim to make solution to a hard problem ”too easy”.
- Similarity between between the two efforts depends on the efficency of the reduction

Setup


Signature scheme and public key
Solution to a hard problem
Signing training
Educated forgery
Malice
Simulated random oracle
Simon

Non-adaptive attack

- A triplet version of El Gamal is used
- No signing training needed
- Simon operates as simulated random oracle for H() queries

First lot of runs

- *1/Adv(k)* runs needed by Malice
- Simon maintains list of e = H(M, r) delivered to Malice
- When Malice outputs a forgery, he has queried the corresponding e

Second lot of runs

- Malice re-runs *1/Adv(k)* times
- Simon resets his RO-answers
- Because on birthday-paradox, two signature pairs (M, (r,e,s)) and (M’, (r’,e’,s’)) satisfy (M,r) = (M’,r’) after number of tries

Extraction of discrete logarithm

$y^r r^s = g^e (\mathrm{mod}\ p)$, $y^r r^{s'} = g^{e'} (\mathrm{mod}\ p)$

$\Leftrightarrow$ $xr + ls = e\ (\mathrm{mod}\ q)$, $xr + ls' = e'\ (\mathrm{mod}\ q)$

$\Leftrightarrow l = (e - e')/(s - s')\ (\mathrm{mod}\ q)$

$x = (e - ls)/r\ (\mathrm{mod}\ q)$

Simon does not care of Malice's method, but is able to extract discrete logarithm.

Reduction results

- Simon's advantage Adv'(k) = 1 / ($q_h^{0,5}$)
- Simon's time cost t' = 2(t+q_h)/Adv(k)
  t is the time Malice needs for a forgery
- This works only if Malice does not care he is fooled

Adaptive chosen-message attack

- Simon simulates also signing of the messages, but does not posses the signing key. Though signature can be verified!
- For signing query, Simon returns:
  $r = g^u y^v \pmod{p}$
  $s = -rv^{-1} \pmod{p - 1}$
  $e = -ruv^{-1} \pmod{p - 1}$
  , where u and v are random integers

ACM-attack results

$t'(k) = 2 * (t(k) + q_H * \tau) + OB(q_s * k^3)/Adv(k)$

$Adv'(k) = q_H^{-0,5}$

- q_s is number of signing queries
- q_H is number of hash-queries
- τ is time consumed in answering a query

Discussion

- The proof suggests that vulnerable parts of this kind of signature are discrete logarithm and the hash-function
- Reduction should run $q_H^{0,5}$ times, which makes the total time $O(q_H^{3/2} / Adv)$
- Mao suggests 2^{50} hash queries
  => $O(2^{75} / Adv)$

Heavy-Row technique

- Created for zero-knowledge identification, but applies to El Gamal also
- Matrix-based approach featuring Malice and Simon
- Two forged signatures lead to contradiction as in forking technique.

Conclusion

- It is possible to reduct from forgering a triplet El Gamal signature to solving discrete logarithm in constant time.
- Using Simon the Simulator offers a good tool to build the reduction.