Chameleon Hashing and Signatures

Hugo Krawczyk
Tal Rabin

IBM T.J. Watson Research Center

Digital Signatures

- **Q:** What's GOOD about digital signatures?

  A: Undeniable commitment to the contents of a document (verifiable by <u>any</u> third party)

  - non-repudiation: judge, certificates, ...

- **Q:** What's BAD about digital signatures?

  A: Undeniable commitment to the contents of a document (verifiable by <u>any</u> third party)

  - unauthorized disclosure of documents: competitors, journalists, ...

    e.g.: confidential contracts, bids, loans, etc

Controlled Verification of Signatures

Conflicting requirements:

- Prevent disclosure to unauthorized parties
- Be able to prove to a judge (to settle disputes)

Q: Possible?

A: Yes, if verification requires signer's action [CvA'89]

Undeniable Signatures

- [CvA'89]: undeniable signatures
- interactive confirmation and denial protocols
  - valid signature $\Rightarrow$ signer cannot deny it
  - forged signature $\Rightarrow$ signer can disprove it

  but proofs are non-transferable
- Many solutions in crypto literature:
  - all based on zero-knowledge protocols
  - overhead relative to regular signatures: computation, communication, interaction
  - extra cryptographic assumptions

Today: Chameleon Signatures

- an efficient alternative to undeniable signatures
- simpler, cheaper, uses standard assumptions
- essentially non-interactive
- totally new approach:
  departs from zero-knowledge proofs
- standard hash-and-sign approach with
  - regular digital signatures (RSA, DSS)
  - special hash functions: *chameleon hashing*

RSA (*chameleon-hash* (message))

Basic Idea

- Allow recipient R to forge at will
  - given a signature of S on m, R can generate signatures of S on *any* other message m'
  - analogy: cut-and-paste of human signatures

  $\Rightarrow$ non-transferability! (who will believe R?)

- What's the value of such signatures?

  Great, if the following hold:
  - signer (and only signer) can prove forgeries
  - signer cannot deny real signatures

  $\Rightarrow$ unforgeability and non-repudiation!

- Technical tool: **chameleon hashing**

“Cut-and-Paste attack”

XYZ Ltd. will supply 30 workstations Heptium-NNY to Crooks Corp. between January and August 1999. *John XYZ*

  John XYZ

XYZ Ltd. will invest 30 million dollars in Crooks Corp. between January and June 1999. *John XYZ*

Reminder: collision-resistant hashing

- no one can find two messages that are hashed to the same value
- instead of SIG(m) can do SIG(HASH(m))
- resistance to collisions preserves unforgeability and non-repudiation

Note:

- if signer can find collisions then it can deny signatures
- if recipient can find collisions then it can forge signatures

Chameleon hash functions

- hash functions with *trapdoor*:
  - collision resistant
  - but: *trapdoor* allows to find collisions!

- Application to chameleon signatures:

  $$\text{SIGN}(\text{CHAM-HASH}(m))$$

  - recipient has trapdoor: can find collisions
    $\Rightarrow$ can forge signatures (non-transferability)
  - signer does not know trapdoor
    $\Rightarrow$ committed to signature (non-repudiation)

Chameleon hashing: closer look

- trapdoor allows to map *any* message to *any* hash value
- how? multi-valued randomized function:
  $$\text{CHAM-HASH}(m, r)$$
- chameleon property (using trapdoor):
  for any m, r, m', can find r' such that
  $$\text{CHAM-HASH}(m, r) = \text{CHAM-HASH}(m', r')$$
- Application to chameleon signatures:
  $$\text{SIGN}(\text{CHAM-HASH}(m))$$
  - hides the value of m
  - allows recipient to forge *any* message (note the name 'chameleon')

  $\Rightarrow$ perfect non-transferability

Chameleon Signatures

- Functions:
  - signature function SIGN$_S$
  - chameleon hash function CHAM-HASH$_R$

- Public keys:
  - verification key for SIGN$_S$
  - computation of hash CHAM-HASH$_R$

- Secret keys:
  - S: signature key for SIGN$_S$
  - R: trapdoor for CHAM-HASH$_R$
    (collision-finding key)

Chameleon Signatures (cont.)

- Signature (from S to R):

$$m, r, \text{SIGN}_S(\text{CHAM-HASH}_R(m, r))$$

- Verification (by R):
  - compute $c = \text{CHAM-HASH}_R(m, r)$
  - verify $\text{SIGN}_S(c)$

- Denial by S:
  - Idea: a false accusation by R allows S to find collisions in CHAM-HASH_R thus proving the fogery!

Proving forgeries

- R provides judge J with triple
  $\widehat{m},\ \widehat{r},\ \mathrm{SIGN}_S(\mathrm{CHAM\text{-}HASH}_R(\widehat{m},\widehat{r}))$
- J verifies signature (computes hash and verifies S's signature on it)
- If verification succeeds: J summons S to deny the triple $\widehat{m},\ \widehat{r},\ \mathrm{SIGN}_S(\mathrm{CHAM\text{-}HASH}_R(\widehat{m},\widehat{r}))$
- If signature is a forgery, S denies it by presenting a collision in $\mathrm{CHAM\text{-}HASH}_R$: the pairs $(\widehat{m},\widehat{r})$ and (m,r) map to the same value!
- Why? Since SIGN_S is unforgeable then there must be a pair (m,r) used by S to produce the original string $\mathrm{SIGN}_S(\mathrm{CHAM\text{-}HASH}_R(m,r))$.

Exposure freeness

- avoid disclosing real signed message during denial

⇒ additional requirement: at denial the signer can present a collision using an arbitrary m' (not necessarily the signed one)

- we achieve this property in a strong sense (signer can choose collisions at will)

Implementation of Chameleon Hashing

- based on standard cryptographic assumptions
- some examples
  - hardness of discrete log:

    $\text{CHAM-HASH}_R(m, r) = g^m y^r \bmod p$

    trapdoor: x such that $y = g^x \bmod p$
  - hardness of factoring:

    $\text{CHAM-HASH}_R(m, r) = 4^m (r^2)^{2^{|m|}} \bmod n$

    trapdoor: primes p, q s.t. $n = p \cdot q$
  - more general assumption:

    trapdoor claw-free pairs
- Note: equivalent to non-interactive chameleon commitment [BCC88].

  Our constructions based on [BKK90, GMR88].

Discrete-log based Chameleon Hashing

Definition:

- primes p, q, $p = kq + 1$; g of order q in Z_p^*
- trapdoor: $x \in Z_q^*$; public key: $y = g^x \bmod p$
- hash: given m, choose random $r \in Z_q^*$ and compute $\text{CHAM-HASH}_y(m, r) = g^m y^r \bmod p$
- collision: m, r, m', r' with $g^m y^r = g^{m'} y^{r'} (\bmod p)$

Properties:

- collision resistance:
  finding collisions $\Rightarrow$ computing disc-log (of y)
  $g^m y^r = g^{m'} y^{r'} \bmod p \Rightarrow x = \frac{m-m'}{r'-r} \bmod q$.
- chameleon trapdoor: given m, r, m' can find r' as $r' = \frac{m+xr-m'}{x} \bmod q$
- denial: given collision (m, r) and $(\widehat{m}, \widehat{r})$ can find $x = \frac{m-\widehat{m}}{\widehat{r}-r} \bmod q$ and with x can find collision with any other m' (*exposure free*).

Further Issues

- recipient-specific nature: main difference with traditional undeniable signatures
- recipient's identity revealed (but can be hidden using "undeniable certificates")
- storage of m, r: with signer or with recipient
- transmission of m, r, sig: unauthenticated link
- convertibility (into regular digital signatures): simple ways to achieve selective and complete convertibility

Summary and Conclusions

- introduced chameleon hashing and signatures
- cryptography can do more than just mimic the "old pen-and-paper world"
- can achieve simultaneously

  + non-repudiation
  + non-transferability
  + unforgeability
  + deniability

- no significant cost beyond digital signatures
  - no interaction, no complex protocols
  - preserves hash-and-sign paradigm
  - computation: less than twice a regular sig.
  - assumptions: as regular sig. (**RSA**, **DSS**)
- yet some room for improvements:
  - avoid recipient-specific nature
  - hide identity of recipient (and that S signed something for R)