Cryptanalytic Attacks on MIFARE Classic Protocol

Jovan Golić

jovan.golic@it.telecomitalia.it

Security Lab, Telecom Italia IT

Outline

1. MIFARE Classic Smart Card
2. Description of MIFARE Classic Protocol
3. Attack Scenarios
4. Objectives
5. Attacks on Genuine Session or Reader
6. Attacks on Genuine Tag – General Concepts
7. Attacks on Genuine Tag – Comparison
8. Generic Differential Attack on Genuine Tag
9. Multiple Sector Attack
10. Conclusions

1. MIFARE Classic Smart Card (1/2)

- **MIFARE Classic smart card is claimed to be the most widely used contactless smart card in the world, especially for access control to buildings and public transport.** It is said to cover more than 70% of market share for access control worldwide.
- It is used in RFID (Radio Frequency IDentification) and NFC (Near Field Communication) systems.
- It implements a proprietary symmetric-key mutual authentication protocol based on a proprietary stream cipher known as CRYPTO1 (1994).
- CRYPTO1 is used for the authentication protocol and also for encrypting messages on RFID/NFC channel between card and reader (without data integrity).

1. MIFARE Classic Smart Card (2/2)

- CRYPTO1 and the protocol are reverse engineered in [Nohl et al. ‘08], [Garcia et al. ‘08] and analyzed in [Nohl et al. ‘08], [de Koning Gans et al. ‘08], [Garcia et al. ‘08], [Garcia et al. ‘09], [Courtois ’09].
- CRYPTO1 uses a preshared 48-bit secret key. On a standard CPU, brute-force attack on 48-bit key can take several years, but less than an hour on FPGA board (e.g., COPACOBANA [Kumar et al. ‘06]).
- **Proposed cryptanalytic attacks in various scenarios demonstrate that MIFARE Classic smart card does not offer 48-bit security level.**
- Dedicated reader infrastructure is not easy to change.

2. Description of MIFARE Classic Protocol (1/6)

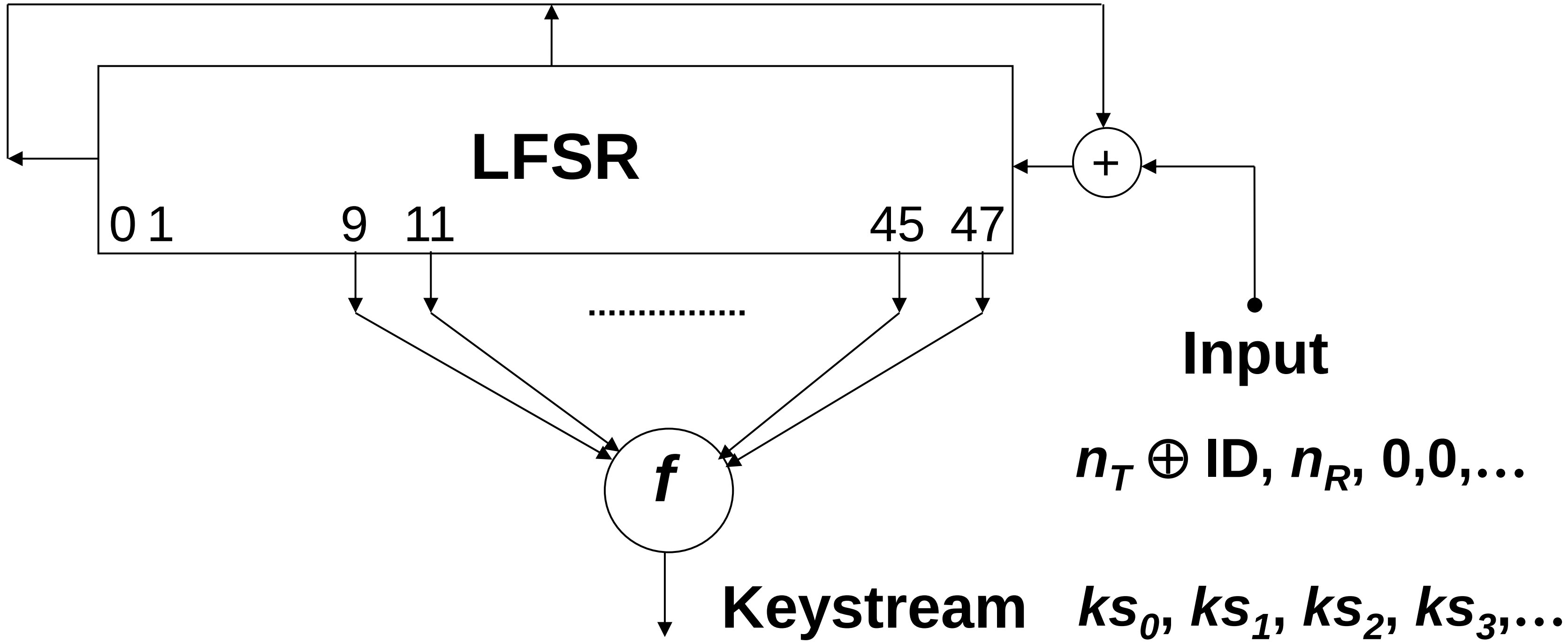
- Memory of MIFARE Classic card/tag is divided into sectors containing16-byte blocks; the last block of each sector contains 48-bit secret key (in fact, two of them); the first block of the first sector contains read-only data including 32-bit tag identifier ID and manufacturer data.
- To write or read data from a given memory block, tag and reader have to be authenticated to each other by a **challenge-response symmetric-key authentication protocol** using the key of the corresponding sector and 32-bit challenge tag and reader nonces n_T and n_R, which are generated by respective pseudorandom number generators.

2. Description of MIFARE Classic Protocol (2/6)

- Tag nonce, n_T, is generated by a 16-bit linear feedback shift register (LFSR), which starts from the same state after powering up the tag and has period 618 ms
    - At most 16 bits of entropy
    - True randomness – variable generation time of tag nonce.
- Reader nonce n_R is generated by a pseudorandom number generator, which starts from the same state after reader restart and generates a nonce upon invocation by authentication protocol
    - True randomness – variable number of invocations after restart.
- **Fresh nonces protect against replay attacks**.

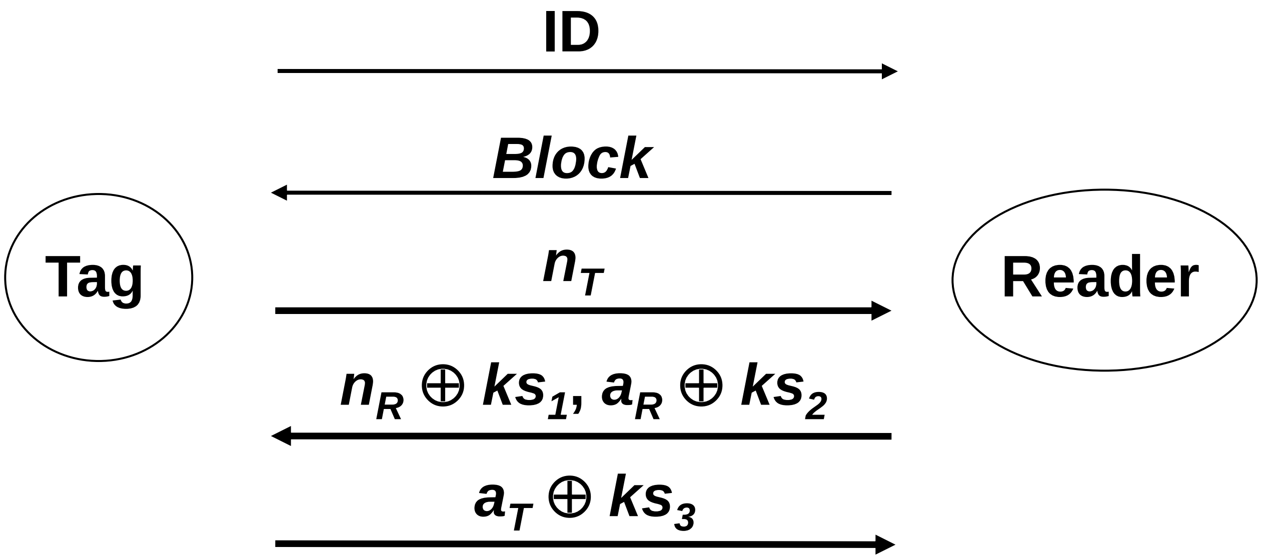
2. Description of MIFARE Classic Protocol (3/6)

- CRYPTO1 – nonlinear filter generator → 48-bit LFSR, 20-bit nonlinear filter function *f*
  - Initialized by secret key and then refreshed by nonces n_T and n_R.



2. Description of MIFARE Classic Protocol (5/6)

- Three-pass authentication protocol for one sector



Tag

Reader

ID

Block

n_T

$n_R \oplus ks_1, a_R \oplus ks_2$

$a_T \oplus ks_3$

$a_R = \mathrm{suc}^2(n_T)$

$a_T = \mathrm{suc}^3(n_T)$

- For multiple sectors, for each new sector, ***Block*** is encrypted by using previous sector key and n_T is sent encrypted with new key as $n_T \oplus ks_0$.

2. Description of MIFARE Classic Protocol (4/6)

- **suc** function maps 32 bits into next 32 bits of 16-bit LFSR.
- First $n_T \oplus$ ID and then n_R are bitwise XORed into LFSR
    - ks_0 is bitwise generated while XORing $n_T \oplus$ ID
    - ks_1 is bitwise generated while XORing n_R
    - ks_2, ks_3, and subsequent keystream are generated by clocking LFSR autonomously.
- Tag and reader generate the same keystream, clocking LFSR forwards, with a difference that reader uses n_R directly, whereas tag first decrypts encrypted n_R bit-by-bit, as the keystream bit encrypting a bit of n_R depends only on previous bits of n_R.
- For multiple sectors, n_T is treated analogously.

2. Description of MIFARE Classic Protocol (6/6)

- **Error detection in MIFARE Classic communication protocol**
  - Every plaintext byte is followed by one parity bit computed as the binary complement of the XOR of all 8 bits **(ISO/IEC 14443-A)**
  - Each parity bit is encrypted with the same keystream bit used to encrypt the next bit of plaintext.
- Due to linear coding preceding the encryption, ciphertext reveals linear relations among keystream bits (1 relation over 8+1 keystream bits per 8+1 ciphertext bits).
- Due to reused keystream bits, ciphertext reveals linear relations among plaintext bits (1 relation over 8+1 plaintext bits per 1+1 ciphertext bits).

3. Attack Scenarios

- Passive, genuine session scenario
  - Attacker intercepts genuine session and aims at decrypting some recorded traces (possibly also reconstructing the key)
  - Impractical, due to very short distances needed.
- Active, genuine reader scenario
  - Attacker uses fake/emulated tag to initiate fake authentication sessions with genuine reader, in order to reconstruct the key
  - Impractical, due to very short distances needed.
- Active, genuine tag scenario, also called tag-only scenario
  - Attacker uses fake/emulated reader to initiate fake authentication sessions with genuine tag (**queries**), in order to reconstruct the key
  - **Easy to implement, provided that the on-line stage is short**.

4. Objectives

- Present a critical comprehensive survey of currently known attacks on MIFARE Classic, in various attack scenarios, and put them into the right perspective in light of the prior art in cryptanalysis.
- Propose improvements of known attacks, if possible.
- **In particular, in tag-only scenario, significantly reduce required on-line time, while keeping off-line time practical**
  - Few seconds or less is very practical, a few tens of seconds is much less practical, a few minutes is impractical, ten or more minutes is very impractical.
- Discuss design drawbacks and design principles.

5. Attacks on Genuine Session or Reader (1/2)

- Objective of two attacks [Garcia et al. ‘08] is to reconstruct the 48-bit key from known keystream
  - 64 bits of ks_2 and ks_3 from genuine session, passive scenario
  - 32 bits of ks_2 and, possibly, additional 32 bits of ks_3 from genuine reader (encrypting known command, such as ‘halt’).
- Internal state, e.g., 48-bit LFSR state S_{64} from which the first bit of ks_2 is generated is first recovered and, then, the key is reconstructed by clocking LFSR backwards (LFSR rollback)
  - Possible even if encrypted $\{n_R\}=n_R \oplus ks_1$ is known instead of n_R
  - Not only for f that does not depend on the left-most LFSR state bit (by branching [Golić et al. ‘00]).

5. Attacks on Genuine Session or Reader (2/2)

- Time-memory-data tradeoff (TMDT) attack in genuine reader scenario requires multiple fake authentication sessions (e.g., 4096 ≈ 10 min) in on-line stage ($T \cdot M = 2^{48}$)
  - Adaptation – success probability equal to 1 instead of being high
  - On-line time can be reduced while keeping off-line time and memory, at the expense of increasing pre-computation time.
- Inversion attack in genuine session or reader scenario requires one/two genuine or fake authentication sessions in on-line stage, for 64/32 known keystream bits
  - Adaptation of generic inversion attack and decimation technique from [Golić '96], [Golić et al. '00], to deal with short keystream
  - Bad tap positions (attack takes $2^{17}/2^{20}$ steps instead of 2^{47} steps).

6. Attacks on Genuine Tag – General Concepts (1/2)

- Attacker can obtain known keystream only because of **a peculiar property of the authentication protocol**
  - In the second pass, fake reader sends fake 64-bit ciphertext, i.e., encrypted values $\{n_R\}$ and $\{a_R\}$, and 8 encrypted parity bits p, $\{p\}$, of n_R and a_R, without knowing the key
  - Authentication by tag will fail, but, if all 8 decrypted bits in p are correct, tag sends back 4-bit ciphertext of a fixed 4-bit error message, encrypted with first 4 bits of ks_3, which reveals 4 keystream bits **(such a query is called successful)**
  - In addition, correct parity bits reveal 8 independent linear relations among keystream bits, i.e., 8 keystream parity bits
  - **Altogether, each sussessful query reveals 12 bits of information about the key**.

6. Attacks on Genuine Tag – General Concepts (2/2)

- Queries can use random or fixed tag nonces. According to [Garcia et al. ‘09], attacker can perform about
  - 1500 queries/sec with a random tag nonce, q_r
  - 30 queries/sec with a fixed tag nonce, q_f, by reset query technique (switch-off the field, power-up passive tag, and start a query at a fixed time after reset; at most ten attempts to get the same n_T).
- **Query strategies in on-line stage:**
  - For random n_T, if $\{n_R\}$, $\{a_R\}$, $\{p\}$ are randomly chosen, then, on average, 256 queries q_r are required to get one successful query, since p is then random (here $\{a_R\}=a_R \oplus ks_2$)
  - For fixed n_T, if $\{n_R\}$, $\{a_R\}$ are fixed and $\{p\}$ is varied, then, on average, 128 queries q_f are required to get one successful query, since p is then fixed.

7. Attacks on Genuine Tag – Comparison (1/2)

- Three attacks proposed in [Garcia et al. ‘09] are here denoted as Att. 1, 2, and 3
  - Att. 1 has impractical off-line time (brute force)
  - Att. 2 has impractical on-line time (15 min)
  - Att. 3 has impractical setup time (brute force).
- The best previously known attack is differential attack proposed in [Courtois ‘09] here denoted as Att. 4
  - Att. 4 makes better use of differential properties of *f* than Att. 2, thus significantly reducing on-line time
  - **In this work, Att. 4 is corrected&optimized into Att. 4***.
- New differential attack, Att. 5, significantly reduces on-line time of Att. 4/4*, while keeping off-line time practical.

7. Attacks on Genuine Tag – Comparison (2/2)

Attacks	**Att. 1**	**Att. 2**	**Att. 3**	**Att. 4***	**New Att. 5**
Setup time	0	0	**2^{48}**	0	0
Setup memory	0	0	$48\cdot 2^{36}$ bit 384 GByte	0	0
On-line time	$1280q_r$ 1 sec	**$28500q_f$** **15 min**	$4230q_r+128q_f$ 7 sec	**$3(256q_r+112q_f)$** **11.7 sec**	**$2(256q_r+48q_f)$** **3.5 sec**
Off-line time	**$5\cdot 2^{48}$** **3 year**	$2^{32.8}$ 10 min	2^{24} TLU 20 sec	$2^{16}+2^{26}f_{ev}$ ≈0	$2^{32}+2^{25}f_{ev}$ 5 min
Off-line memory	≈0	≈0	≈0	168 Byte	42 KByte
Success Prob	100%	100%	100%	99.4%	99.1%

8. Generic Differential Attack on Genuine Tag (1/5)

- **Objective:**
  - From a number of successful queries for different (n_T, {n_R}), recover a set of candidate LFSR states at a given time, for some (n_T, {n_R})
  - By LFSR rollback, then recover a set of candidate keys and, finally, get the unique key possibly using additional successful queries
  - **Problem: the values of n_R are unknown to the attacker**.
- **First on-line phase:** Get one successful query (random n_T).
- **Second on-line phase:** Modify last m bits of {n_R} and then fix n_T, {n_R}, {a_R} and vary last 5 bits of {p} until a successful query occurs; on average, 16 such queries with a fixed n_T, for each of 2^m-1 modifications of {n_R} are needed.
- On-line stage thus yields 2^m successful queries.

8. Generic Differential Attack on Genuine Tag (2/5)

- Since $n_R = \{n_R\} \oplus ks_1$, where the last m bits of ks_1 depend on the last m bits of $\{n_R\}$, each m-bit change δ_m of $\{n_R\}$ can result in the same change of n_R with some probability depending on the filter function f.
- This is the probability that the last m keystream bits of ks_1, used for encrypting the last m bits of $\{n_R\}$, are independent of the last m bits of $\{n_R\}$
  - The first of these m bits is already independent, as it depends on previous bits of $\{n_R\}$ only
  - The remaining m-1 bits are output bits of (m-1)-bit augmented filter function of all LFSR state bits as input bits; these m-1 output bits are independent of the last m-1 input bits with probability π_{m-1}

8. Generic Differential Attack on Genuine Tag (3/5)

  - It follows that $\pi_0=1$, $\pi_1=29/32\approx0.906$, $\pi_2=(29/32)^2\approx0.821$
  - Worst design case: $\pi_{m-1}=1$ would hold if *f* were independent of the last *m* LFSR state bits
  - Best design case: $\pi_{m-1}=0$ would hold if *f* were linear in the last LFSR state bit (a sufficient condition from [Golić '96]).
- Since LFSR sequence depends linearly on n_R, for each value of δ_m, subsequent LFSR sequence can be expressed as bitwise XOR of unknown LFSR sequence corresponding to initial value of $\{n_R\}$ and a known binary sequence determined by known value of δ_m **(this holds with probability π_{m-1})**.
- 4 keystream bits from each successful query are used.

8. Generic Differential Attack on Genuine Tag (4/5)

- Attacker thus obtains $4\cdot 2^m$ equations of the form $z_i=f(S_i\oplus\Delta_i)$, for 2^m values of δ_m and $96\leq i\leq 99$, where S_i is unknown LFSR state for $\delta_m=0$ at time i, and z_i and Δ_i depend only on δ_m (to be solved in off-line stage).
- Nonlinear system of equations can be solved for S_{99} by an adapted variant of the well-known resynchronization attack [Daemen et al. '94] (essentially, exhaustive search) and using the fact that
  - f depends only on 20 bits of LFSR state and after 2 steps, f depends on the same 19 bits and 1 new bit (bad tap positions)
  - Accordingly, z_{96} and z_{98} depend on 21 LFSR bits of S_{99} (as well as z_{97} and z_{99} depend on other 21 LFSR bits of S_{99}).

8. Generic Differential Attack on Genuine Tag (5/5)

- Att. 4* is corrected&optimized Att. 4 from [Courtois '09], which is a differential attack for m=3
  - Queries with random/fixed tag nonces are better handled
  - For one run, success probability $\pi_2 \approx 0.821$ (instead of $\approx$0.75) and average on-line time $\approx$3.9 sec (instead of $\approx$8.52 sec for first run and $\approx$5.33 sec for other runs), and off-line time $\approx$0
  - For 3 independent runs of Att. 4*, success probability $\approx$0.994, average on-line time $\approx$11.7 sec, and average off-line time $\approx$0.
- **Att. 5 is a new differential attack, for *m*=2**
  - For one run, success probability $\pi_1 \approx 0.906$, average on-line time $\approx$1.77 sec, and off-line time $\approx$5 min
  - For 2 independent runs of Att. 5, success probability $\approx$0.991, average on-line time $\approx$3.5 sec, and average off-line time $\approx$5 min.

9. Multiple Sector Attack

- **Assumption:** Key for the first sector has been already recovered in any described attack scenario.
- Main feature is that n_T is sent encrypted by tag
  - This should prevent known keystream attacks, unless
  - **Entropy of n_T is small!** (Repeated keystream bits, timing.)
- By effectively guessing n_T, attacker gets 96 keystream bits in genuine session scenario [Garcia et al. ‘08] and 32 keystream bits in tag-only scenario [Garcia et al. ‘09] and then applies adapted inversion attack.
- **Consequently, tag-only attack scenario is much easier for other sectors than for the first sector**.

10. Conclusions (1/3)

- **MIFARE Classic protocol does not offer 48-bit security level due to:**
  - Repeatable and predictable tag nonces
  - Really bad tap positions for filter function and bad choice of filter function in CRYPTO1
  - Generation&encryption of parity bits for error detection
  - Peculiar feature that tag can sent an encrypted response if authentication of reader fails.
- The most effective known attack is adapted inversion attack in genuine session or genuine reader scenarios
  - Impractical, due to very short distances to genuine static reader required.

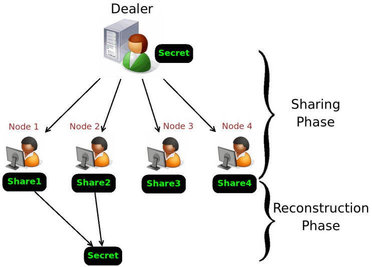
10. Conclusions (2/3)

- The most practical attack scenario is genuine tag scenario, and the new attack, requiring only about 3 sec of on-line time, is the most effective currently known attack in this scenario
  - It significantly improves on the most effective previously known attack, requiring about 10 sec of on-line time.
- **Practical countermeasures**
  - Use electromagnetic-shield covers for tags on smart cards
  - User notification or activation on demand of on-going NFC communications, for tags on mobile devices.
- **Impractical countermeasure**: Replace MIFARE Classic cards/tags and dedicated readers (costly).

10. Conclusions (3/3)

- **Light-weight cryptography**
  - Ease of implementation
  - Reduced, but guaranteed security level
  - Designs secure against known attacks
  - Reduced security margins with respect to known attacks
  - **Difficult to satisfy in practice.**
- **Obscurity in cryptography**
  - Unknown algorithms make cryptanalysis significantly more difficult, if not impossible
  - Designs should be secure against known attacks even if algorithms are compromised (e.g., reverse engineered)
  - **Typically not satisfied in practice.**

Asynchronous Computational VSS with Reduced Communication Complexity



Cryptographer's Track, RSA Conference 2013

Michael Backes
Saarland University & MPI-SWS
Germany

Amit Datta
Carnegie Mellon University
USA

Aniket Kate
Saarland University
Germany

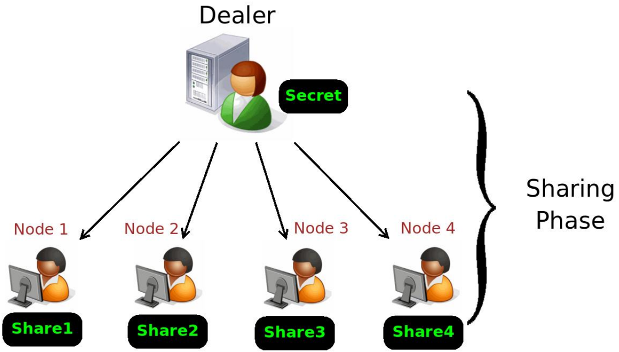
Outline

- Background
  - Secret Sharing
  - Asynchronous Verifiable Secret Sharing (AVSS)
- State-of-the-art for AVSS and Shortcomings
- Our Protocols
  - eAVSS: efficient AVSS
  - eAVSS-SC: efficient AVSS with Strong Commitment

Secret Sharing

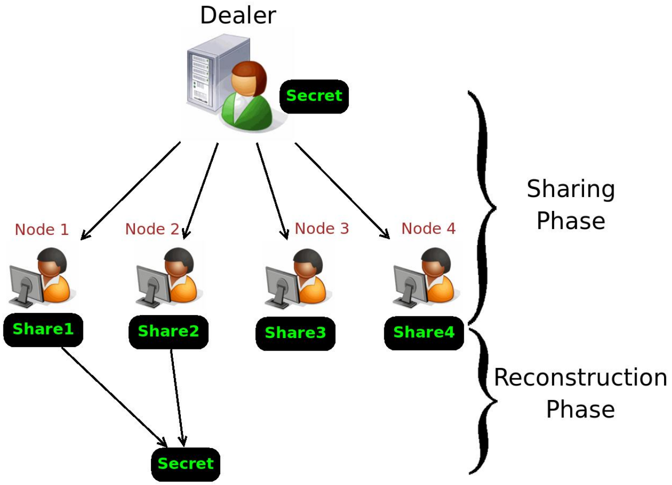
Secret Sharing

Sharing Phase



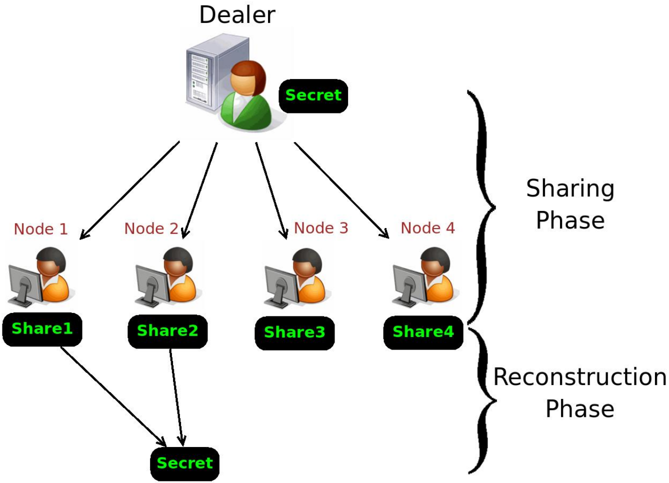
Secret Sharing

Reconstruction Phase



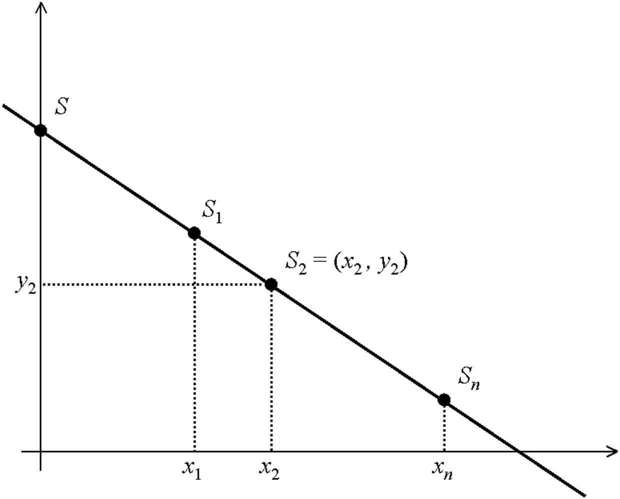
Secret Sharing

$(4, 1)$-Secret Sharing



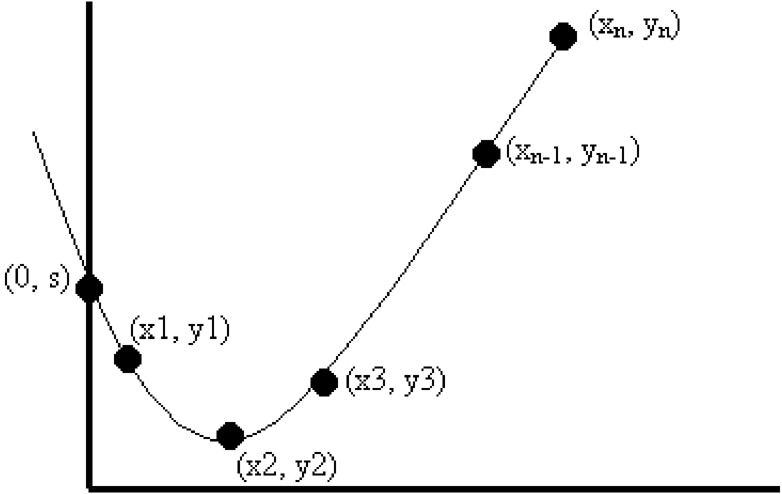
Polynomial-Based Shamir Secret Sharing

$(n, 1)$-Secret Sharing for secret S



Polynomial-Based Shamir Secret Sharing

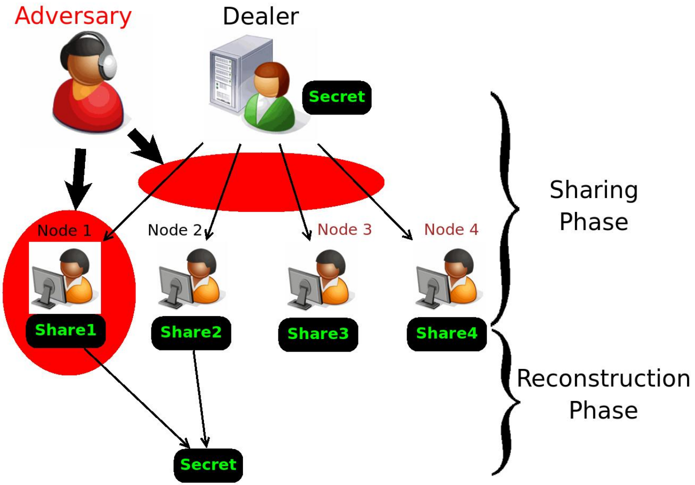
Not an $(n, 1)$-Secret Sharing for secret S



Verifiable Secret Sharing—VSS

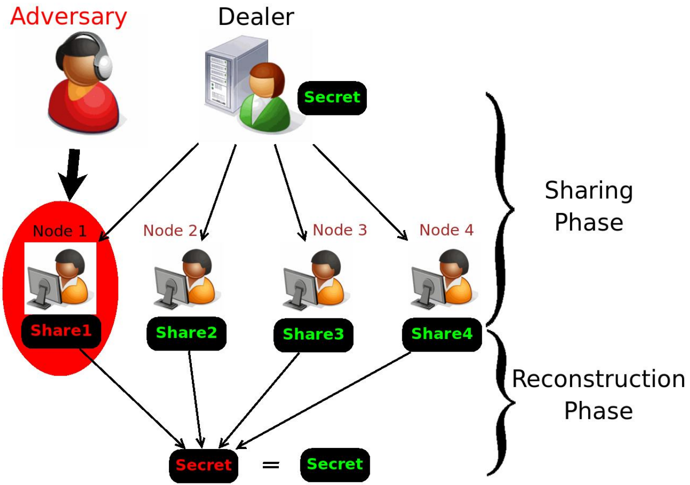
Verifiable Secret Sharing—VSS

Privacy Property



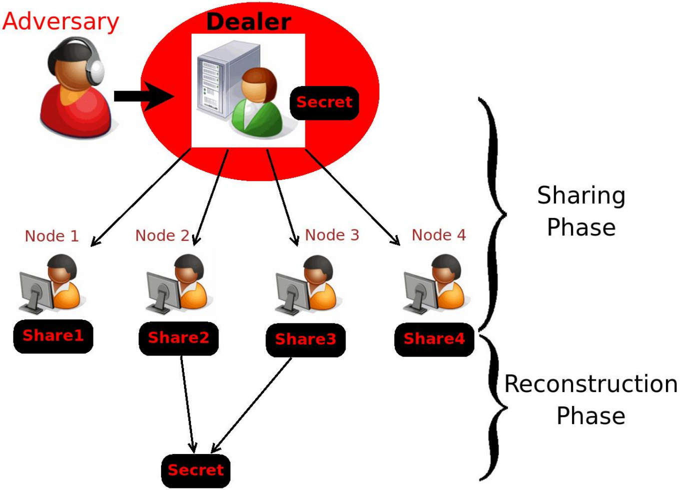
Verifiable Secret Sharing—VSS

Correctness Property



Verifiable Secret Sharing—VSS

Commitment Property



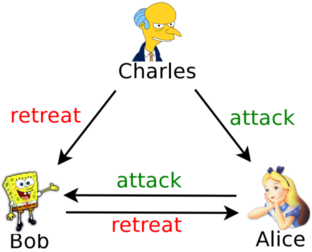
VSS in the Literature

Applications

1. Multi-party Computation
2. Threshold Cryptography

$$\lambda_i = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

3. Byzantine Agreement



4. Coin-tossing

VSS in the Literature

Communication Assumption

Most of the VSS protocols in the literature assume that the network to be synchronous. However,

- the Internet does not always function synchronously, and
- there is no reliable broadcast channel available for free.

VSS in the Literature

Communication Assumption

Most of the VSS protocols in the literature assume that the network to be synchronous. However,

- the Internet does not always function synchronously, and
- there is no reliable broadcast channel available for free.

Therefore, we need VSS protocols that work in the asynchronous communication setting

VSS in the Literature

Communication Assumption

Most of the VSS protocols in the literature assume that the network to be synchronous. However,

- the Internet does not always function synchronously, and
- there is no reliable broadcast channel available for free.

Therefore, we need VSS protocols that work in the asynchronous communication setting

State-of-the-art

- Asynchronous VSS (AVSS) protocols are inefficient in terms of communication complexity
- The best known AVSS protocol [Cachin et al., ACM CCS ’02] communicates $O(\kappa n^3)$ bits

Our Contributions

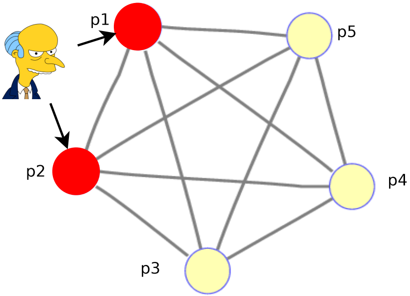
- Communication complexity for AVSS can be reduced using the concept of (constant-size) polynomial commitments
- We present two AVSS protocols with $O(\kappa n^2)$ communication complexity:
  - Our first protocol satisfies the standard VSS definition
    Applications:
    1. Stand-alone VSS scenarios
    2. Byzantine agreement
  - Our second protocol satisfies a stronger VSS definition
    Applications:
    1. multiparty computation (MPC)
    2. threshold cryptography

Outline

- Background
  - Secret Sharing
  - Asynchronous Verifiable Secret Sharing (AVSS)
- State-of-the-art for AVSS and Shortcomings
- Our Protocols
  - eAVSS: efficient AVSS
  - eAVSS-SC: efficient AVSS with Strong Commitment

Asynchronous Message Passing Setting

Asynchronous Message Passing Setting



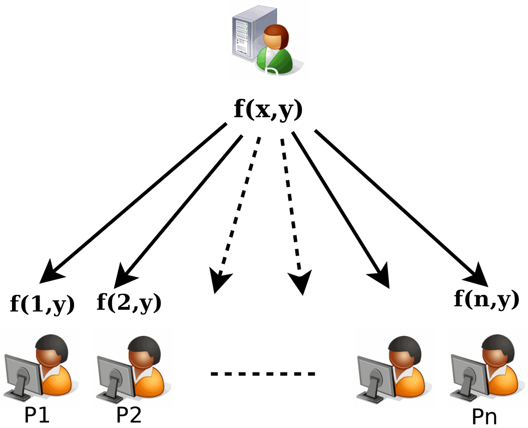
processes: $p_1, \ldots, p_n$

- pairwise connected by an asynchronous channel
    - messages can be arbitrarily delayed, or reordered
    - however, messages are eventually delivered

- at most t of n processes may exhibit faulty behavior In this setting, the optimal resiliency bound is $n \geq 3t + 1$

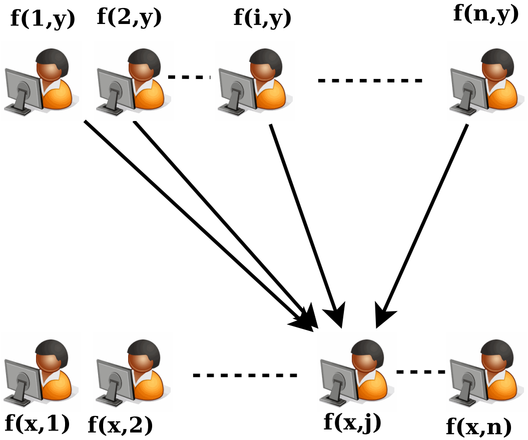
State-of-the-art Protocol for AVSS

Sharing Phase



State-of-the-art Protocol for AVSS

Sharing Phase



State-of-the-art Protocol for AVSS

Message complexity: $O(n^2)$

For verification:

$$\text{Commitment matrix } \mathbf{C} = \{C_{jl}\} = g^{f_{jl}}$$

State-of-the-art Protocol for AVSS

Message complexity: $O(n^2)$

For verification:

Commitment matrix $\mathbf{C} = \{C_{jl}\} = g^{f_{jl}}$

Reduced from $O(n^2)$ to $O(n)$ with hash functions.

State-of-the-art Protocol for AVSS

Message complexity: $O(n^2)$

For verification:

Commitment matrix $\mathbf{C} = \{C_{jl}\} = g^{f_{jl}}$

Reduced from $O(n^2)$ to $O(n)$ with hash functions.

Communication complexity: $O(\kappa n^3)$,

where κ is the security parameter

Reference :

C. Cachin, K. Kursawe, A.Lysyanskaya, and R. Strobl.
Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems.
In ACM CCS ’02.

Outline

- Background
    - Secret Sharing
    - Asynchronous Verifiable Secret Sharing (AVSS)
- State-of-the-art for AVSS and Shortcomings
- Our Protocols
    - eAVSS: efficient AVSS
    - eAVSS-SC: efficient AVSS with Strong Commitment

What We Want to Achieve

Message complexity: $O(n^2)$,

Communication complexity: $O(\kappa n^2)$,

where κ is the security parameter

What We Want to Achieve

Message complexity: $O(n^2)$,

Communication complexity: $O(\kappa n^2)$,

where κ is the security parameter

Solution : Constant-size Polynomial Commitments
Helps commit to a univariate polynimial by publishing just one value

Reference:
A.Kate, G. M. Zaverucha, and I. Goldberg.
Constant-Size Commitments to Polynomials and Their Applications.
In Proceedings of ASIACRYPT '10.

A PolyCommit scheme

Setup$(1^\kappa, t)$ generates an appropriate algebraic structure $\mathcal{G} = \langle e, \mathbb{G}, \mathbb{G}_T \rangle$ and the system parameter PK

Commit$(PK, f(x))$ outputs a commitment $\mathcal{C}$ to a polynomial $f(x)$

CreateWitness$(PK, f(x), i)$ outputs $\langle i, f(i), w_i \rangle$, where w_i is a witness for the evaluation $f(i)$ of $f(x)$

VerifyEval$(PK, \mathcal{C}, i, f(i), w_i)$ verifies that $f(i)$ is indeed the evaluation of the polynomial committed in $\mathcal{C}$

A PolyCommit scheme

Setup$(1^\kappa, t)$ generates an appropriate algebraic structure $\mathcal{G} = \langle e, \mathbb{G}, \mathbb{G}_T \rangle$ and the system parameter PK

Commit$(PK, f(x))$ outputs a commitment $\mathcal{C}$ to a polynomial $f(x)$

CreateWitness$(PK, f(x), i)$ outputs $\langle i, f(i), w_i \rangle$, where w_i is a witness for the evaluation $f(i)$ of $f(x)$

VerifyEval$(PK, \mathcal{C}, i, f(i), w_i)$ verifies that $f(i)$ is indeed the evaluation of the polynomial committed in $\mathcal{C}$

Construction

- Single element commitments for univariate polynomials
- However, the scheme does not work for multi-variate polynomials required in AVSS schemes!!

Our eAVSS Protocol

Dealer D

- Select a polynomial $f(x)$, such that $f(0) = s$
- $\mathcal{C} = \text{Commit}(PK, f(x))$, $w_i = \text{CreateWitness}((PK, f(x), i))$
- Send ($\mathcal{C}$, w_i, $f(i)$) to every party P_i

Our eAVSS Protocol

Dealer D

- Select a polynomial $f(x)$, such that $f(0) = s$
- $\mathcal{C} =$ Commit$(PK, f(x))$, $w_i =$ CreateWitness$((PK, f(x), i))$
- Send ($\mathcal{C}$, w_i, $f(i)$) to every party P_i

Party P_i

- If VerifyEval(PK, $\mathcal{C}$, i, $f(i)$, w_i) succeeds, send (`echo`, $\mathcal{C}$)
- On receiving $(n - t)$ (`echo`, $\mathcal{C}$), send (`ready`, `share`, $\mathcal{C}$)
- Otherwise:
  - (a) On receiving $(n - 2t)$ (`ready`,*, $\mathcal{C}$) signals, send (`ready`, `share`, $\mathcal{C}$) to every party P_j.
  - (b) On receiving $(n - 2t)$ (`ready`,*, $\mathcal{C}'$) signals, send (`ready`, `no-share`, $\mathcal{C}'$) to every party P_j.
- On receiving $(n - t)$ (`ready`, $\mathcal{C}$) signals, and at least $(n - 2t)$ contain `share`, terminate

Salient Points

- There are at least $n - 2t \geq 3t + 1 - 2t = t + 1$ honest parties with correct shares

- There are at most n `send`, n^2 `echo`, and n^2 `ready` messages

Properties of eAVSS

Liveness. If the dealer D is honest, then all honest parties complete sharing.

Secrecy. If D is honest, then the adversary has no information about s.

Agreement. If some honest party completes the sharing phase, then all honest parties will eventually complete the sharing phase.

Properties of eAVSS

Liveness. If the dealer D is honest, then all honest parties complete sharing.

Secrecy. If D is honest, then the adversary has no information about s.

Agreement. If some honest party completes the sharing phase, then all honest parties will eventually complete the sharing phase.

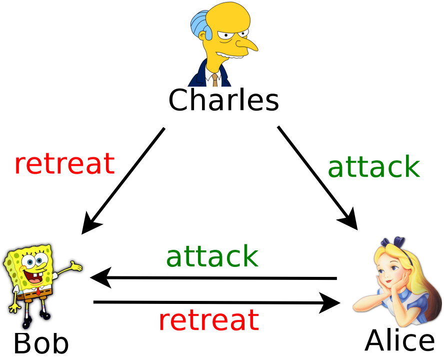
Correctness. Once all honest parties complete sharing, there exists a fixed value $z \in \mathbb{Z}_p$, such that the following holds:

(a) If an honest dealer has shared the secret s, then $s = z$.

(b) If each of the honest servers P_i reconstructs some z_i, then $z_i = z$

Suitable for

Byzantine Agreement



Charles

retreat

attack

attack

retreat

Bob

Alice

Coin-tossing

Not Suitable for

Multi-party Computation

Threshold Cryptography

$$\lambda_i = \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}$$

Properties of Stronger AVSS

Properties of Stronger AVSS

Liveness. If the dealer D is honest, then all honest parties complete sharing.

Secrecy. If D is honest, then the adversary has no information about s.

Agreement. If some honest party completes the sharing phase, then all honest parties will eventually complete the sharing phase.

Strong Correctness. Once $t + 1$ honest parties complete sharing, there exists a fixed value $z \in \mathbb{Z}_p$, such that the following holds:

(a) If an honest dealer has shared the secret s, then $s = z$.

(b) If each of the honest servers P_i reconstructs some z_i, then $z_i = z$

Protocol for eAVSS-SC

- Dealer `sends` polynomials $f^0(x), f^1(x), ... f^n(x)$, with $f^k(x) = F(x, k)$, $F(x, y)$ is of degree $\leq t$. Commitments: $\mathcal{C}^0, \mathcal{C}^1, ..., \mathcal{C}^n$.

Protocol for eAVSS-SC

- Dealer `sends` polynomials $f^0(x), f^1(x), ... f^n(x)$, with $f^k(x) = F(x,k)$, $F(x,y)$ is of degree $\leq t$. Commitments: $\mathcal{C}^0, \mathcal{C}^1, ..., \mathcal{C}^n$.

- There will be at least $t+1$ honest parties with correct polynomials $f^k(x) = F(x,k)$, and they compute their shares $s_k = f^k(0) = F(0,k) = F(k,0) = f^0(k)$

Protocol for eAVSS-SC

- Dealer `sends` polynomials $f^0(x), f^1(x), ... f^n(x)$, with $f^k(x) = F(x,k)$, $F(x,y)$ is of degree $\leq t$. Commitments: $\mathcal{C}^0, \mathcal{C}^1, ..., \mathcal{C}^n$.

- There will be at least $t+1$ honest parties with correct polynomials $f^k(x) = F(x,k)$, and they compute their shares $s_k = f^k(0) = F(0,k) = F(k,0) = f^0(k)$

- These $t+1$ parties can enable any P_i to reconstruct its polynomial $f^i(x)$ by sending $f^k(i) = f^i(k)$

Protocol for eAVSS-SC

- Dealer `sends` polynomials $f^0(x), f^1(x), ...f^n(x)$, with $f^k(x) = F(x,k)$, $F(x,y)$ is of degree $\leq t$. Commitments: $\mathcal{C}^0, \mathcal{C}^1, ..., \mathcal{C}^n$.
- There will be at least $t+1$ honest parties with correct polynomials $f^k(x) = F(x,k)$, and they compute their shares $s_k = f^k(0) = F(0,k) = F(k,0) = f^0(k)$
- These $t+1$ parties can enable any P_i to reconstruct its polynomial $f^i(x)$ by sending $f^k(i) = f^i(k)$

Problem: Have to send a vector of commitments in the `echo` and `ready` messages.

Solution: Perform another round of PolyCommit on hash values of the commitments.

Take Away

- Constant-size polynomial commitments help to obtain an AVSS protocol with reduced communication complexity

- We have presented two efficient AVSS schemes (eAVSS and eAVSS-SC) with $O(n^2\kappa)$ communication complexity
- They reduce the communication complexity of various AVSS applications by a linear (in n) factor

Take Away

- Constant-size polynomial commitments help to obtain an AVSS protocol with reduced communication complexity

- We have presented two efficient AVSS schemes (eAVSS and eAVSS-SC) with $O(n^2\kappa)$ communication complexity
- They reduce the communication complexity of various AVSS applications by a linear (in n) factor

Thanks!