Breaking a practical cipher in the real world by solving multivariate polynomial systems

Chen-Mou Cheng

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
ccheng@cc.ee.ntu.edu.tw

March 3, 2013

Outline

How we got started on attacking MIFARE Classic

- May, 2009: Read "Wirelessly Pickpocketing a Mifare Classic Card"
- Summer, 2009: Repeated the experiments on Taipei EasyCard
- Fall, 2009: Demonstrated several attacks to the authority
  - Card-only attacks (Nijmegen)
  - Long-range sniffing (new)
- Jan., 2010: Government regulators *approved* EasyCard as a means of electronic payment in Taiwan
- Sep., 2011: First EasyCard hacking incident reported in media
  - Suspect promptly arrested
  - Soon the authority disclosed upgrade plans to "EasyCard 2.0," claiming that it will be "secure"
- Aug., 2012: Official release of EasyCard 2.0

What is EasyCard?

- A contactless smartcard for payment in Taipei public transportation systems since 2002
- Also accepted in numerous convenient stores, drug stores, restaurants, cafes, supermarkets, book stores, movie theaters, . . . , since 2010
- More than 35 million issued; more than 4.7 million transactions daily

Reverse-engineering a real-world RFID payment system

- A talk by Harald Welte in 27C3, Dec., 2010
- Disclosed "the process of reverse-engineering the actual content of the [EasyCard] to discover the public transportation transaction log, the account balance and how the daily spending limit work"
- As well as "how easy it is to add or subtract monetary value to/from the card. Cards manipulated as described in the talk have been accepted by the payment system"
- "Corporations enabling citizens to print digital money"

The "secure" EasyCard 2.0

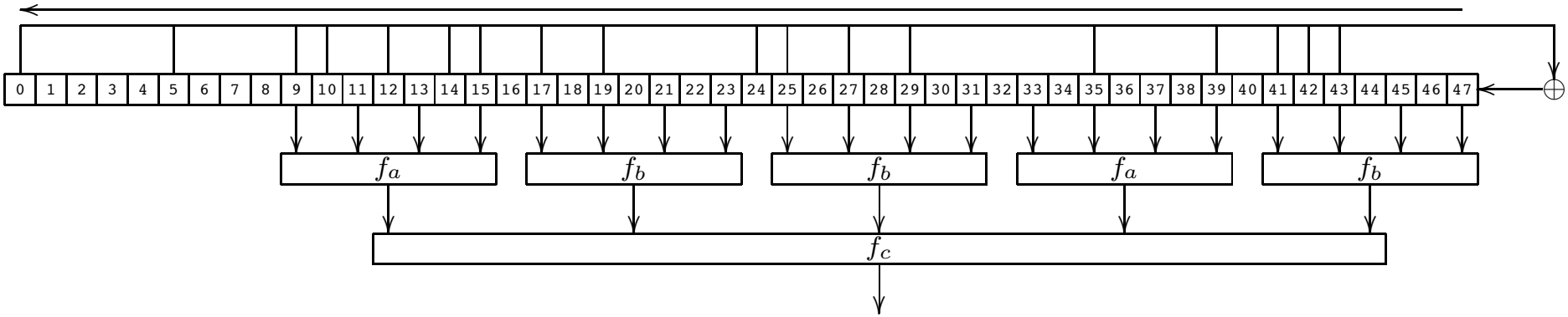
- Recall: Weaknesses of MIFARE Classic
  - Parity weaknesses
  - Nested authentications
- EasyCard 2.0 still uses MIFARE Classic, but:
  - Tag replies with `0x0` error code whether parities are correct or not
  - Tag nonce now is unpredictable and seems to have 32-bit entropy, disabling attacks based on tag nonce manipulation and nested authentications
- Sure, sniffing still works if you have a legitimate reader
  - So does brute-force if you don't have such a reader, which may take years on an ordinary PC
- *All* other existing, efficient card-only attacks no longer work
  - Seems "secure" enough from a practical point of view

The research question

- Really?
- More specifically, is there a practically relevant card-only attack on EasyCard 2.0?

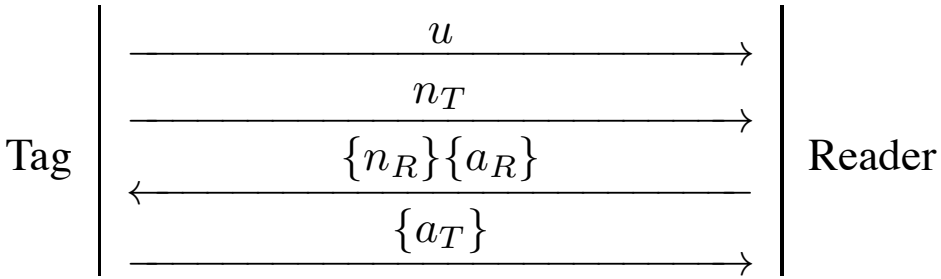
Outline

The structure of Crypto-1



- 48-bit LFSR
- Nonlinear filter function $f = f_c(f_a, f_b, f_b, f_a, f_b)$
  - $\deg f_a = \deg f_b = 3$
  - $\deg f_c = 4$
  - Therefore, $\deg f = 12$

The use of Crypto-1 in MIFARE Classic



- $u \oplus n_T$ is used to initialize the cipher
- It then got further initialized by encrypted $\{n_R\}$
- After $\{n_R\}$, the cipher stops taking input and keeps outputting keystream bits
- Except that for EasyCard 2.0, it would respond with (encrypted) `0x0` error code if there is anything wrong

First attempt: Algebraic attack

- An algebraic model for the cipher
  - Treat initial LFSR state (key) as unknown $\mathbf{x} = (x_0, \ldots, x_{47})$
  - An input bit i will produce new LFSR state $\mathbf{A}_i(\mathbf{x}) = \mathbf{L}\mathbf{x} + \mathbf{v}_i$
  - Similarly, a sequence of n input bits $\mathbf{i}$ will produce new LFSR state $\mathbf{A}_\mathbf{i}(\mathbf{x}) = \mathbf{L}^n\mathbf{x} + \mathbf{v}_\mathbf{i}$ for some $\mathbf{v}_\mathbf{i}$ that depends on $\mathbf{i} = u \oplus n_T, n_R, \mathbf{0}, \ldots$
  - Output keystream bit is $y = f(\mathbf{x}_\mathbf{i})$ given LSFR state $\mathbf{x}_\mathbf{i}$
- EasyCard 2.0 still gives out 4 keystream bits after a failed authentication attempt
- Idea: Collect some (≥ 12) traces and solve the resulting system of nonlinear equations using Gröbner-basis or SAT solvers
- Unfortunately this *does not work* because n_R results in NLFSR and hence equations of saturating degrees

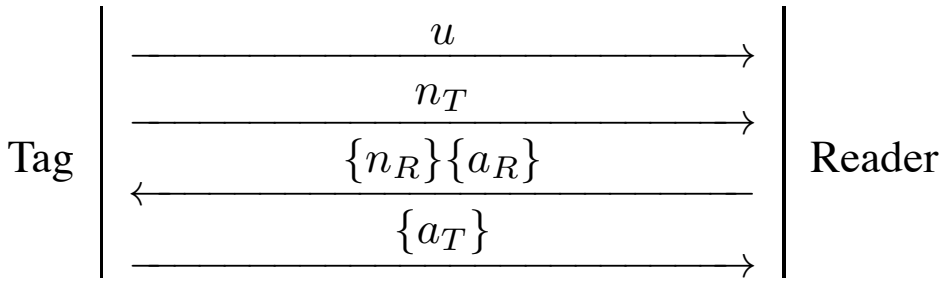
Outline

Acknowledgment

- M. Albrecht and C. Cid: "Algebraic techniques in differential cryptanalysis" (FSE 2009)
- S. Knellwolf, W. Meier, and M. Naya-Plasencia: "Conditional differential cryptanalysis of NLFSR-based cryptosystems" (ASIACRYPT 2010)

LFSR differences

- Recall that a sequence of n input bits $\mathbf{i}$ will produce new LFSR state $\mathbf{A_i}(\mathbf{x}) = \mathbf{L}^n\mathbf{x} + \mathbf{v_i}$ for some $\mathbf{v_i}$ that depends on $\mathbf{i}$
- The difference of two LFSR states that descend from a common ancestor is $\mathbf{A_i}(\mathbf{x}) \oplus \mathbf{A_j}(\mathbf{x}) = \mathbf{v_i} \oplus \mathbf{v_j}$, which *does not depend on* $\mathbf{x}$

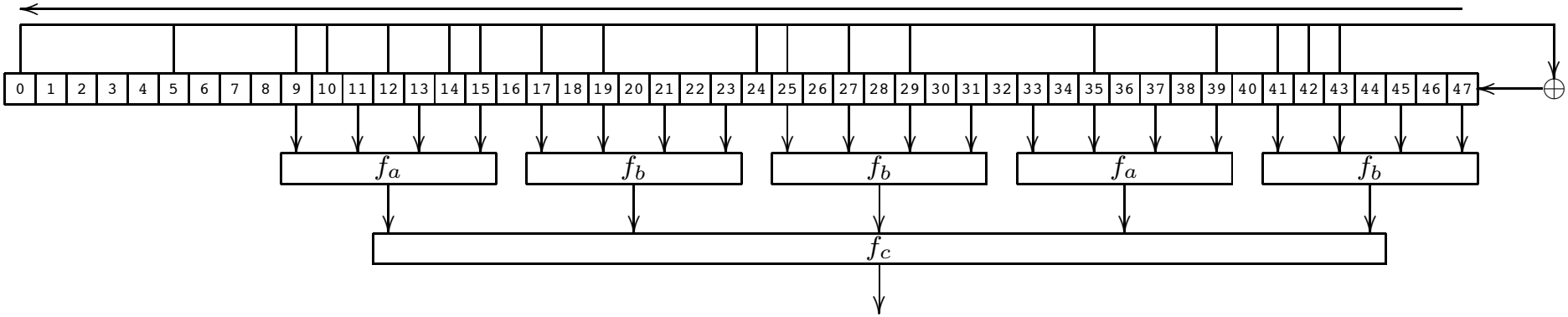


- Therefore, we can *know* the LFSR state difference after two different n_T's, even though we *cannot control* it
  - Just be patient: Wait long enough, and good things will happen
  - Fortunately, birthday paradox significantly reduces the wait time

Differential attack

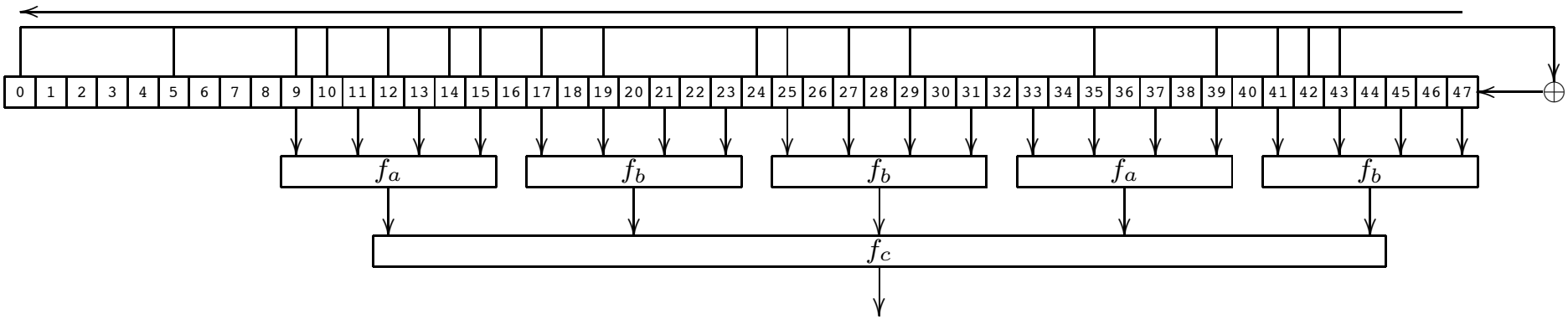
- A differential is a pair of input and output differences $(\Delta x, \Delta y)$, where $\Delta y = S(x \oplus \Delta x) \oplus S(x)$
  - Δx is easy to get from repeated authentications (birthday paradox)
  - Δy is not so easy
  - But we can always guess!
  - If we are right, we can cancel it by manipulating $\{n_R\}$
- Idea: Can use the 4 keystream bits after a failed authentication attempt as an *oracle* for our earlier guesses of Δy's, a special form of conditional differential cryptanalysis

Some nice example input difference



- Recall that we know the LFSR state difference after two different n_T's
- State difference $\Delta x_9 = 1$ (and 0 elsewhere) could be "cancelled" easily by manipulating LFSR states via $\{n_R\}$, assuming we can correctly "guess" the difference in output keystream bits

Some nice example input difference we can get



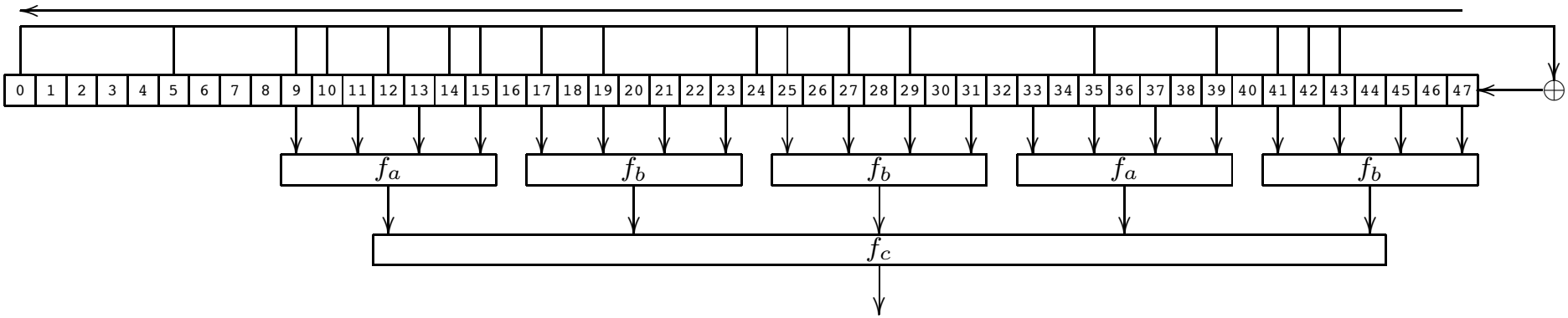
- Unfortunately the difference cannot happen because $|n_T| = 32$
- In practice, the best input difference is $\Delta x_{16} = 1$
  - Experiments show that the probability of 4 correct guesses is $1/6$

Algebraic differential attack

- For each correct guess, we get a (degree-11) equation $f(\mathbf{x} \oplus \Delta\mathbf{x}) \oplus f(\mathbf{x}) = \Delta y$, where Δy is our guess of the difference
- Idea: Collect some more traces and solve the resulting system of nonlinear equations using Gröbner-basis or SAT solvers
- Unfortunately this *does not work so well*, possibly because there are a lot of redundancies in the resulted system of equations

Our key observation

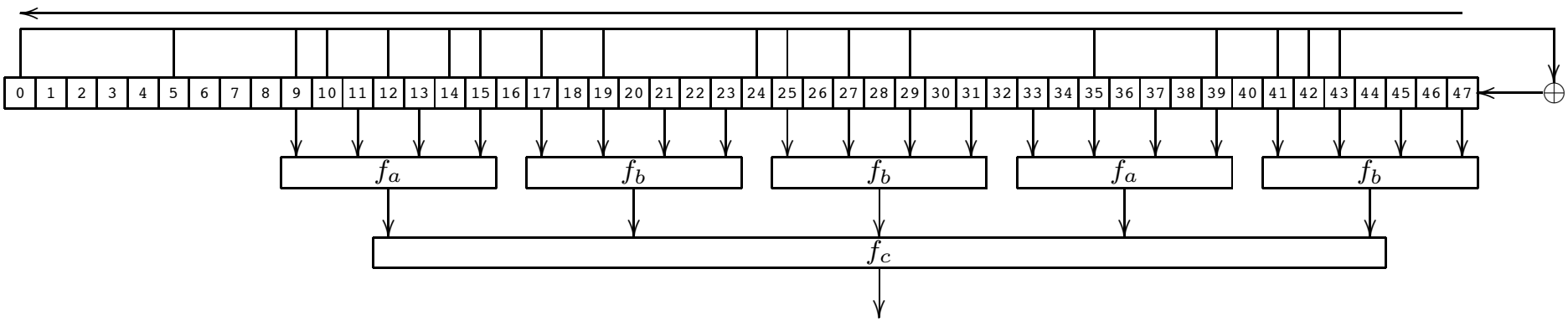
- For the input difference $\Delta x_{16} = 1$, what happens when the second bit of $\{n_R\}$ is being fed in?



- What is the LFSR state at this moment?
- Call this state $\mathbf{z} = (z_0, \ldots, z_{47})$
  - $\mathbf{z}$ is *almost* an image of the key $\mathbf{x}$ under some affine transformation $\tilde{A}_{n_T}$ that depends on n_T (and the fixed u)
  - By "almost" we mean only z_{47} is "contaminated" by nonlinearity
  - Note that Δz_{47} is still zero

What does this mean?

- We know that $\mathbf{z}$ is a solution to our difference equation $\frac{\partial f}{\partial z_{15}} = \Delta y$
- We can also express $z_0, \ldots, z_{46}$ as linear combinations of the unknown $\mathbf{x}$ (and the known $u \oplus n_T$)
- We don't know z_{47}, but it can only be 0 or 1
- However, not all assignments of $(z_9, z_{11}, \ldots, z_{47})$ are possible
- They need to satisfy the equation $(\frac{\partial f}{\partial z_{15}} \oplus \Delta y)(\frac{\partial^2 f}{\partial z_{15} \partial z_{47}} \oplus 1) = 0$



Obtaining hint bits

- As a result, each trace can act like a filter, eliminating about 1/4 of the solution space
- In reality, there are dependencies among the traces, so we would need more (about 60) traces
- At the end, we can then obtain a few (≤ 16) candidate assignments that pass all filters, which will help tremendously in Gröbner-basis or SAT solving

Outline

Experiment setup

- All experiments are performed on an old laptop and a standard ACR 122 reader
  - Running Ubuntu with libraries such as `libnfc` and `crapto1`
- We only report timing numbers for CryptoMiniSat
  - The CNF formulas are generated by our own software
    - ★ Will soon open-source the software
  - We also tried some Gröbner-basis solvers
    - ★ The built-in solver in Maple
    - ★ PolyBoRi
  - Unfortunately the performance is not (yet) on par with SAT solvers

EasyCards under attack

Card type	Parities checked	n_T generation
EasyCard 1.0	Yes	Predictable
EasyCard 1.5	Yes	Somewhat random
EasyCard 2.0	No (always `0x0`)	Random

Experiment results

Attack type	Online time	Compute time	1.0	1.5	2.0
Sniffing attack	2 sec.	$<$ 2 sec.	$\surd$	$\surd$	$\surd$
GPU brute-force	5 sec.	14 hours	$\surd$	$\surd$	$\surd$
CPU brute-force	5 sec.	50 years	$\surd$	$\surd$	$\surd$
Parities attack	$>$ 3 min.	$<$ 30 sec.	$\surd$	?	
Nested authentications	15–75 sec.	25–125 sec.	$\surd$	$\surd$	
This attack	10–20 hours	2–15 min.			$\surd$

Possible fixes

- Turn off the oracle!
  - Why give the attacker information when it's not necessary?
- Perhaps this would break some existing readers
- In this case, can increase $|n_T|$ to, say, 64 bits
  - Nice differentials would then take forever to show up
- The real fix is to stop using MIFARE Classic, period

Future works, a.k.a. things we should've done but haven't

- I lied about the oracle—it's actually not that good
- In experiments, whenever we see an opportunity to produce the $\Delta x_{16} = 1$ difference, we succeed with a probability of about 23%
  - 6% are false positives due to collision
- Need a way to distinguish them from the other 17% true positives

Some preliminary ideas

- The equation $(\frac{\partial f}{\partial z_{15}} \oplus \Delta y)(\frac{\partial^2 f}{\partial z_{15} \partial z_{47}} \oplus 1) = 0$ is a *necessary condition* for true positives and can be used to remove some false positives
- Similarly, can use the following as a *sufficient condition* to identify some true positives

$$\frac{\partial f'}{\partial z_k} = \Delta y_k \text{ and } \frac{\partial^2 f'}{\partial z_k \partial x} = 0, k = 9, 11, 13, 15,$$

  where $f'(z_9, \ldots, z_{39}, x) = f_c(f_a(z_9, \ldots, z_{15}), f_b(z_{17}, \ldots, z_{23}), \ldots, x)$
- We can then mix these known true positives with some unknowns in Gröbner-basis or SAT solvers
  - Can disjunctively combine several unknowns to increase the probability that the resulting equation holds

Real future works

- The Nijmegen people told us that the data collection time should be much, much shorter (maybe 1/10)
  - Need to figure out how to control the reader better
- Try to incorporate the power of Gröbner-basis solvers!
  - For example, they should be useful to detect the presence of false positives because the Gröbner basis of such a system is $\{1\}$ and hence should come out quickly in the computation
  - Or they can be combined with SAT solvers, hopefully yielding more powerful system solvers
- Try to apply similar techniques to attack other ciphers
  - The key idea is to encode the conditions for certain differentials to happen in algebraic equations

Thank you!

- Questions or comments?