Bypassing Smart-Card Authentication and Blocking Debiting:

Vulnerabilities in Atmel Cryptomemory based Stored-Value Systems

> Jonathan Lee Neil Pahl

Who are we?

Jonathan Lee – UBC Computer Engineering student

Neil Pahl – UBC Electrical Engineering, recent graduate

Disclaimer !

- We are not (yet) security industry professionals or 1337 code-breaking hackers

- We did not break Smart Card Security

 But, we can show that even with introductory security knowledge, we were able to find vulnerability in a 'secure' system's implementation

What do we know about Security?

EECE 412: Intro to Computer Security

- Fundamental Security Principles
- Cryptography
- Authentication
- Access Control
- Secure Design

What Was Our Motivation For this Project?

Term Project to Perform a Security Analysis

> (Opportunity to Legitimately try Cool Hacks)

- Smart Cards are synonymous with Security
- We hoped to emulate known replay attacks on a nearby Smart Card laundry system

Why should anyone care about our Hack?

- Companies spend time and money to add more security functions to their products
- However, security functions cannot protect the system if they are not implemented thoughtfully

History?

The need for Authentication + Encryption

- Joe Grand (2009) SF parking meters, vulnerable to replay attack
- Strom Carlson (2006) Fedex Kinko's SLE4442, vulnerable to simple password replay

Smart card laundry machine (stored value card)



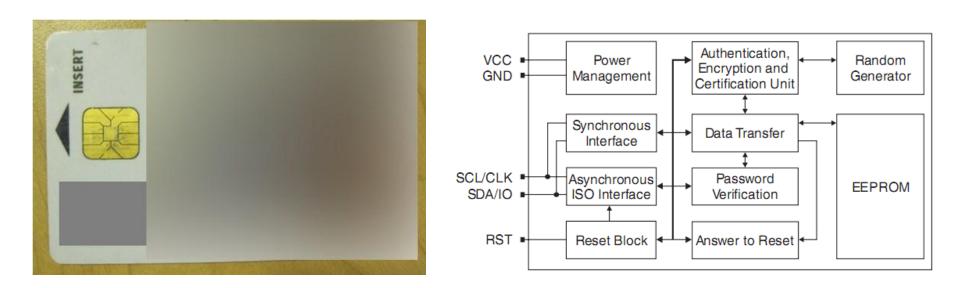
We were thinking of trying some of the usual attacks:

- Could we simply read and modify monetary values directly on the card?
- Could we sniff out read/write passwords?
- Could we perform replay attacks?

But, the success of these attacks depend on the security measures in place

- A few Google searches lead us to an interesting blog of someone attempting to hack a laundry system (http://tinyurl.com/2csk6dr)
 - Their success was limited to reading the configuration settings from the card.
 - We couldn't confirm it yet, but we decided to start with the assumption that our system used the same chip.

Atmel CryptoMemory AT88SC0404 Smart-card ATR: 3B B2 11 00 10 80 00 04



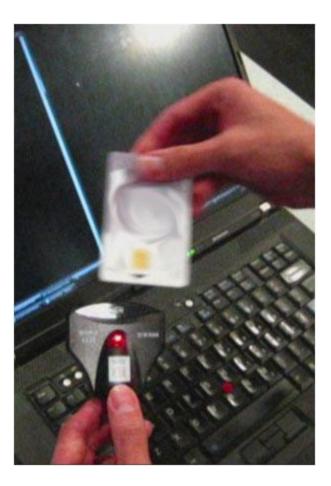
High Security Features:

- 64-bit Mutual Authentication Protocol (under license of ELVA)
- Encrypted Checksum
- Stream Encryption
- Four Key Sets for Authentication and Encryption
- Eight Sets of Two 24-bit Passwords
- Anti-tearing Function
- Voltage and Frequency Monitor

Security Modes:

Mode	Configuration Data	User Data	Passwords	Data Integrity Check
Standard/Password	clear	clear	clear	n/a
Authentication	clear	clear	encrypted	MAC
Encryption	clear	encrypted	encrypted	MAC

- Standard/password mode is Default
- Command sequences are needed to enter other security modes





\$0 \$8 \$10	3b	b2	11	00	10	80	00	04	90 90 90	00 00 00
\$18 \$20 \$28 \$30 \$38 \$40	bf df ff ff ff	08 ff ff ff	df ff ff ff	08 ff ff ff	df ff ff ff	58 ff ff ff	df ff ff ff	58 ff ff ff	90 90 90 90 90 90	00 00 00 00 00 00
\$40 \$48	ff	ff	ff	ff	ff	ff	ff	ff	90	00
\$50 \$58	ff 69	e7 00	02	63	79	85	54	8d	90	00
\$60	ff	ff	ff	ff	ff	ff	ff	ff	90	00
\$68 \$70	69 ff	00 ff	ff	ff	ff	ff	ff	ff	90	00
\$78 \$80	69 ff	00 ff	ff	ff	ff	ff	ff	ff	90	00
\$88 \$90	69 69	00								
\$98 \$a0	69 69	00								
\$a8 \$b0 \$b8	69 ff ff	00 20 20	20 20	20 20	ff ff	20 20	20 20	20 20	69 69	00 00
\$c0 \$c8	ff	20 20	20 20	20 20	ff ff	20 20	20 20	20 20	69 69	00
\$d0 \$d8	ff ff	20 20	20 20	20 20	ff ff	20 20	20 20	20 20	69 69	00
\$e0	ff	20	20	20	ff	20	20	20	69	00
\$e8	ff	20	20	20	ff	20	20	20	69	00

Interpreting the Memory Dump:

\$0 \$8 \$10	3b	b2	11	00	10	80	00	04	90 90 90	00 00 00
\$18 \$20 \$28 \$30 \$38 \$40 \$48 \$50	bf df ff ff ff ff	08 ff ff ff ff	df ff ff ff ff 02	08 ff ff ff ff 63	df ff ff ff ff 79	58 ff ff ff ff 85	df ff ff ff ff 54	58 ff ff ff ff 8d	90 90 90 90 90 90 90 90 90	00 00 00 00 00 00 00 00
\$58 \$60	69 ff		ff	ff	ff	ff	ff	ff	90	00
\$68 \$70	69 ff	00 ff	ff	ff	ff	ff	ff	ff	90	00
\$78 \$80	69 ff	00 ff	ff	ff	ff	ff	ff	ff	90	00
\$88 \$90 \$98 \$a0	69 69 69 69 69	00 00 00 00								
\$a8 \$b0 \$b8 \$c0 \$c8	ff ff ff ff	00 20 20 20 20 20	20 20 20 20	20 20 20 20	ff ff ff ff	20 20 20 20	20 20 20 20	20 20 20 20	69 69 69 69	00 00 00 00
\$d0 \$d8 \$e0 \$e8	ff ff ff ff	20 20 20 20	20 20 20 20	20 20 20 20	ff ff ff ff	20 20 20 20	20 20 20 20 20	20 20 20 20 20	69 69 69 69	00 00 00 00

Interpreting the Memory Dump:

Addresses

\$0 \$8 \$10 \$28 \$20 \$28 \$30 \$38 \$40 \$38 \$40 \$58 \$50 \$58 \$60 \$58 \$60 \$68 \$70 \$78	bf df ff ff ff 69 ff 69	b2 08 ff ff ff ff ff o0 ff 00 ff 00 ff	df ff ff ff ff ff ff ff ff	08 ff ff ff 63 ff	df ff ff ff ff ff ff	58 ff ff ff ff 85 ff	df ff ff ff ff ff ff	ff	90	00
\$80 \$98 \$90 \$98 \$20 \$28 \$00 \$28 \$20 \$20 \$20 \$20 \$20 \$20 \$20 \$20 \$20 \$20	ff 69 69 69 69 ff ff ff ff ff	ff 00 00 00 20 20 20 20 20 20 20 20 20 20	20 20 20 20 20 20 20 20 20 20		ff ff ff ff ff	ff 20 20 20 20 20 20 20 20 20		20 20 20 20 20 20 20 20 20 20	90 69 69 69 69 69 69 69	00 00 00 00 00 00 00 00 00

Interpreting the Memory Dump:

- Addresses
- Status 'Access Granted'

\$0	3b	b2	11	00	10	80	00	04	90	00
\$8									90	00
\$10									90	00
\$18	bf								90	00
\$20	df	08	df	08	df	58	df	58	90	00
\$28	ff	ff	ff	ff	ff	ff	ff	ff	90	00
\$30	ff	ff	ff	ff	ff	ff	ff	ff	90	00
\$38	ff	ff	ff	ff	ff	ff	ff	ff	90	00
\$40	ff	ff	ff	ff	ff	ff	ff	ff	90	00
\$48	ff	ff	ff	ff	ff	ff	ff	ff	90	00
\$50	ff	e7	02	63	79	85	54	8d	90	00
\$58	69	00								
\$60	ff		ff	ff	ff	ff	ff	ff	90	00
\$68	69	00								
\$70	ff	ff	ff	ff	ff	ff	ff	ff	90	00
\$78	69	00								
\$80	ff		ff	ff	ff	ff	ff	ff	90	00
\$88	69	00								
\$90	69	00								
\$98	69	00								
\$a0	69	00								
\$a8	69	00			~ ~					
\$b0	ff	20	20		ff		20		69	
\$b8	ff	20	20	20	ff	20	20	20	69	00
\$c0	ff	20	20	20	ff	20	20	20	69	00
\$c8	ff	20	20	20	ff	20	20	20	69	00
\$d0	ff	20	20	20	ff	20	20	20	69	00
\$d8	ff	20	20	20	ff	20	20	20	69	00
\$e0	ff	20	20	20	ff	20	20	20	69	00
\$e8	ff	20	20	20	ff	20	20	20	69	00

Interpreting the Memory Dump:

- Addresses
- Status 'Access Granted'
- Status 'Access Denied'

	bf df ff ff ff	b2 08 ff ff ff e7 00 ff	df ff ff ff ff	08 ff ff ff ff 63	df ff ff ff ff 79	58 ff ff ff ff 85	df ff ff ff 54	58 ff ff ff	90 90 90 90 90 90 90 90 90	00 00 00 00 00 00 00 00 00 00
\$70 \$78	69 ff 69 ff	00 ff 00 ff						ff ff	_	00
\$88 \$90 \$98 \$a0 \$a8	69	00 00 00 00 00								
\$b0 \$b8 \$c0 \$c8 \$d0 \$d8 \$e0 \$e8	ff ff ff ff ff ff ff	20 20 20 20 20 20 20 20 20 20	20 20 20 20 20 20 20 20 20	20 20 20 20 20 20 20 20 20	ff ff ff ff ff ff	20 20 20 20 20 20 20 20 20	20 20 20 20 20 20 20 20 20	20 20 20 20 20 20 20 20 20	69 69 69 69 69 69 69 69	00 00 00 00 00 00 00

Interpreting the Memory Dump:

DCR – Data Config Reg

\$8 \$10	b b2	11	00	10	80		04	90 90 90 90	00 00 00 00
\$28 f \$30 f \$38 f	f 68 f ff f ff f ff f ff	df ff ff ff	08 ff ff ff ff	df ff ff ff	58 ff ff ff ff	df ff ff ff	58 ff ff ff	90 90 90 90 90	00 00 00 00 00
	f ff	ff	ff	ff	ff	ff	ff	90	ŏŏ
	f e7	02	63	79	85	54	8d	90	00
\$60 f	9 00 f ff 9 00	ff	ff	ff	ff	ff	ff	90	00
		ff	ff	ff	ff	ff	ff	90	00
\$80 f \$88 6	9 00	ff	ff	ff	ff	ff	ff	90	00
\$98 6 \$a0 6	9 00 9 00 9 00								
	9 00 f 20	20	20	££	20	20	20	60	00
	f 20 f 20	20 20	20 20	ff	20 20	20 20	20 20	69 69	00 00
\$c0 f	f 20	20	20	ff	20	20	20	69	00
	f 20 f 20	20 20	20 20	ff ff	20 20	20 20	20 20	69 69	00
	f 20	20	20	ff	20	20	20	69	00 00
\$e0 f	f 20	20	20	ff	20	20	20	69	00
\$e8 f	f 20	20	20	ff	20	20	20	69	00

Interpreting the Memory Dump:

 DCR – Data Config Reg 1011 1111

\$		3b	b2	11	00	10	80	00	04	90 90	00 00
\$	10 18			df	0.8	df	5.0	df	5.0	90 90	00
	20' 28	ar ff	08 ff	df ff	08 ff	ff	58 ff	ff	58 ff	90 90	00 00
-	30	ff	90	00							
	38	ff ff	ff	ff	ff ff	ff	ff	ff	ff	90	00
	40 48	ff	ff ff	ff ff	ff	ff ff	ff ff	ff ff	ff ff	90 90	00 00
-	50	ff	e7	02		79	85	54	8d	90	00
	58	69	00								
	60	ff		ff	ff	ff	ff	ff	ff	90	00
-	68 70	69 ff	00	ff	ff	ff	ff	ff	ff	00	00
-	78	69	00							90	00
	80	ŤŤ		ff	ff	ff	ff	ff	ff	90	00
-	88	69	00								
-	90	69	00								
	98 a0	69 69	00								
	a0 a8	69	00								
	bÖ	ŤŤ	20	20	20	ff	20	20	20	69	00
-	b8	ff	20	20	20	ff	20	20	20	69	00
	c0	ff	20	20	20	ff	20	20	20	69	00
	c8 d0	ff ff	20 20	20 20	20 20	ff ff	20 20	20 20	20 20	69 69	00 00
-	d8	ff	20	20	20	ff	20	20	20	69	õõ
-	e0	ff	20	20	20	ff	20	20	20	69	00
\$	e8	ff	20	20	20	ff	20	20	20	69	00

Interpreting the Memory Dump:

DCR – Data Config Reg
 1011 1111
 4 Authen. Attempts Allowed

\$8 \$10 \$18 \$20 \$28 \$30 f	of 17 08 f ff	11 df ff	08 ff ff	df ff ff	58 ff ff	df ff ff	58 ff ff	90 90 90 90 90 90 90	00 00 00 00 00 00 00
	f ff f ff	ff ff	ff ff	ff ff	ff ff	ff ff	ff ff	90 90	00 00
	if ff	ff	ff	ff	ff	ff	ff	90	00
	f e7	02		79	85	54	8d	90	ŏŏ
	59 00					-			
		ff	ff	ff	ff	ff	ff	90	00
	59 00	~ ~	~ ~	~ ~	~ ~	~ ~	~ ~	~ ~	
	f ff	ff	ŤΤ	ŤŤ	τŦ	ΤŤ	ŤŤ	90	00
	59 00 f ff	ff	ff	ff	ff	ff	ff	90	00
	59 00	•••	•••		•••	•••	•••	50	00
	59 00								
	59 OO								
	59 OO								
	59 00	20	20	££	20	20	20	60	00
	f 20 f 20	20 20	20	ff ff	20 20	20 20	20 20	69 69	00 00
	f 20	20	20	ff	20	20	20	69	00
	f 20	žõ	žŏ	ff	žŏ	žŏ	žŏ	69	ŏŏ
	f 20	20	20	ff	20	20	20	69	00
	f 20	20	20	ff	20	20	20	69	00
	f 20	20	20	ff	20	20	20	69	00
\$e8 f	f 20	20	20	ff	20	20	20	69	00

Interpreting the Memory Dump:

- DCR Data Config Reg
 1011 1111
 - 4 Authen. Attempts Allowed
 - Authen. Attemps Counter Enabled

\$8 \$10 \$18 \$20 \$28 \$30 \$38 \$40 \$48 \$48 \$50 \$58 \$58 \$58 \$58 \$58 \$69 \$60 \$68 \$68 \$68 \$68	• 08 • ff • ff • ff • ff • e7 • 00 • 00	df ff ff ff ff 02 ff	08 ff ff ff 63 ff	df ff ff ff 79 ff	58 ff ff ff ff 85 ff	df ff ff ff ff 54			
\$70 ff \$78 69	00			ff					
\$80 ff \$88 69 \$90 69 \$98 69 \$a0 69 \$a8 69	 00 00 00 00 00 00 00 00 	ff	ff	ff		ff	ff	90	00
\$b0 ff \$b8 ff	20	20 20	20 20	ff	20	20 20	20 20	69 69	00
\$c0 ff \$c8 ff \$d0 ff	20	20 20 20	20 20 20	ff ff ff	20 20 20	20 20 20	20 20 20	69 69 69	00 00 00
\$d8 ff \$e0 ff \$e8 ff	20 20	20 20 20	20 20	ff ff ff	20 20 20	20 20 20	20 20 20	69 69 69	00 00 00

Interpreting the Memory Dump:

• ARn – Access Registers

\$0 \$8 \$10 \$18 \$20		b2 08	11 df	00 08	10 df	80 58		04 58	90 90 90 90 90	00 00 00 00 00
\$28 \$30	ff	ff	ff	ff	ff	ff	ff	ff ff	90 90	00
\$38 \$40 \$48	ff ff ff	ff ff	ff ff ff	ff ff ff	ff ff ff	ff ff ff	ff ff ff	ff ff ff	90 90 90	00 00 00
\$50 \$58	ff 69	e7 00	02	63	79	85	54	8d	90 90	00
\$60 \$68	ff 69		ff	ff	ff	ff	ff	ff	90	00
\$70 \$78	ff 69	ff 00	ff	ff	ff	ff	ff	ff	90	00
\$80 \$88 \$90 \$98 \$a0 \$a8	ff 69 69 69 69 69		ff	ff	ff	ff	ff	ff	90	00
\$b0 \$b8	ff ff	20 20	20 20	20 20	ff ff ff	20 20	20 20	20 20	69 69	00
\$c0 \$c8 \$d0 \$d8 \$e0 \$e8	ff ff ff ff ff	20 20 20 20 20 20 20	20 20 20 20 20 20 20	20 20 20 20 20 20 20	ff ff ff ff ff	20 20 20 20 20 20 20	20 20 20 20 20 20 20	20 20 20 20 20 20 20	69 69 69 69 69 69	00 00 00 00 00 00

Interpreting the Memory Dump:

ARn – Access Registers
 1101 1111

\$0 \$8	3b	b2	11	00	10	80	00	04	90 90	00
\$10 \$18 \$20	df	80	df	08	df	58	df		90 90 90	00 00 00
\$28 \$30 \$38	ff ff	ff ff	ff ff	ff ff ff	ff ff	ff ff ff	ff ff	ff ff ff	90 90 90	00 00 00
\$40 \$48 \$50	ff ff ff	ff ff e7	ff ff 02	ff ff 63	ff ff 79	ff ff 85	ff ff 54	ff ff 8d	90 90 90	00 00 00
\$58 \$60 \$68	69 ff 69	00	ff		ff					
\$70 \$78	ff 69	ff 00			ff				90	00
\$80 \$88 \$90	ff 69 69	ff 00 00	ff	ff	ff	ff	ff	ff	90	00
\$98 \$a0	69 69	00 00								
\$a8 \$b0 \$b8	69 ff ff	00 20 20	20 20	20 20	ff ff	20 20	20 20	20 20	69 69	00 00
\$c0 \$c8 \$d0	ff ff ff	20 20 20	20 20	20 20 20	ff ff ff	20 20 20	20 20	20 20	69 69	00 00 00
\$d8 \$e0	ff ff	20 20	20 20 20	20 20	ff ff	20 20	20 20 20	20 20 20	69 69 69	00
\$e8	ff	20	20	20	ff	20	20	20	69	00

Interpreting the Memory Dump:

• ARn – Access Registers 1101 1111 Encryption not necessary

\$0 \$8 \$10 \$18 \$20 \$28 \$30 \$38 \$40 \$48 \$50 \$58 \$50 \$58 \$60		08 ff ff ff ff e7 00	df ff ff ff ff 02	08 ff ff ff ff 63	df ff ff ff ff 79	80 58 ff ff ff ff 85 ff	df ff ff ff ff 54	ff ff ff ff 8d	90 90 90 90 90 90 90 90 90 90	00 00 00 00 00 00 00 00 00 00 00
\$68	69	00								
\$70 \$78	ff 69	ff 00	tt	tt	tt	ff	tt	tt	90	00
\$78 \$80 \$88 \$90 \$98 \$a0 \$a8	69 69 69 69 69 69		ff	ff	ff	ff	ff	ff	90	00
\$b0	ff	20	20	20	ff	20	20	20	69	00
\$b8 \$c0	ff ff	20 20	20 20	20 20	ff ff	20 20	20 20	20 20	69 69	00 00
\$c8	ff	20	20	20	ff	20	20	20	69	00
\$e0	ff	20	20	20	ff	20	20	20	69	00
\$d0 \$d8	ff ff ff ff	20 20 20 20	20 20 20 20	20 20	ff ff ff ff	20 20	20 20	20 20	69 69	00 00

Interpreting the Memory Dump:

• ARn – Access Registers
1101 1111
L
Encryption not necessary
no read/write passwords

\$0 \$8 \$10 \$18	3b bf	b2	11	00	10	80	00	04	90 90 90 90	00 00 00 00
\$20 \$28 \$30 \$38		08 ff ff ff	df ff ff	08 ff ff	df ff ff	58 ff ff ff	df ff ff	58 ff ff ff	90 90 90 90	00 00 00 00
\$40 \$48	ff	ff	ff	ff	ff	ff	ff	ff	90 90	00
\$50 \$58	ff 69	e7 00	02		79	85	54	8d	90	00
\$60 \$68	ff 69	ff 00	ff	ff	ff	ff	ff	ff	90	00
\$70 \$78	ff 69	ff 00	ff	ff	ff	ff	ff	ff	90	00
\$80 \$88 \$90 \$98 \$a0 \$a8	ff 69 69 69 69 69	ff 00 00 00 00 00	ff	ff	ff	ff	ff	ff	90	00
\$b0 \$b8 \$c0 \$c8 \$d0	ff ff ff ff	20 20 20 20 20 20	20 20 20 20 20 20	20 20 20 20	ff ff ff ff	20 20 20 20 20	20 20 20 20 20	20 20 20 20 20	69 69 69 69 69	00 00 00 00 00
\$d8 \$e0 \$e8	ff ff ff	20 20 20	20 20 20	20 20 20	ff ff ff	20 20 20	20 20 20	20 20 20	69 69 69	00 00 00

Interpreting the Memory Dump:

• ARn – Access Registers

Authentication required for read/write command use

\$0 \$8 \$10 \$18	3b bf	b2	11	00	10	80	00	04	90 90 90 90	00 00 00 00
\$20 \$28 \$30		08 ff ff	df ff ff	08 ff ff	df ff ff	58 ff ff	df ff ff	58 ff ff	90 90 90	00 00 00
\$38 \$40 \$48	ff ff ff	90 90 90	00 00 00							
\$50 \$58	ff 69	e7 00	02		79	85	54	8d	90 90	00
\$60 \$68	ff 69	00				ff				
\$70 \$78 \$80	ff 69 ff	ff 00 ff				ff ff			90 90	
\$88 \$90	69 69	00 00							50	00
\$98 \$a0 \$a8	69 69 69	00 00 00								
\$b0 \$b8	ff ff	20 20	20 20	20 20	ff	20 20	20 20	20 20	69 69	00 00
\$c0 \$c8 \$d0	ff ff ff	20 20 20	20 20 20	20 20 20	ff ff ff	20 20 20	20 20 20	20 20 20	69 69 69	00 00 00
\$d8 \$e0	ff ff	20 20	20 20	20 20	ff ff	20 20	20 20	20 20	69 69	00
\$e8	ff	20	20	20	ff	20	20	20	69	00

We Now Know:

We Now Know:

- Authentication Attempts Counter Enabled
- 4 Authentication Attempts Allowed ______

What it Tells Us:

Brute force key cracking is impossible

We Now Know:

- Authentication Attempts
 Counter Enabled
- 4 Authentication Attempts Allowed _____

- Brute force key cracking is impossible
- Encryption not necessary Might be able to sniff read/write passwords

We Now Know:

- Authentication Attempts
 Counter Enabled
- 4 Authentication Attempts Allowed _____

- Brute force key cracking is impossible
- Encryption not necessary —>> Might be able to sniff read/write passwords
- No read/write passwords > No Passwords to sniff

We Now Know:

- Authentication Attempts
 Counter Enabled
- 4 Authentication Attempts Allowed _____

- Brute force key cracking is impossible
- Encryption not necessary —>> Might be able to sniff read/write passwords
- No read/write passwords > No Passwords to sniff
- Authentication required for read/write command
- System will be run under Authentication Mode

Recall the Security Modes:

Mode	Configuration Data	User Data	Passwords	Data Integrity Check	
Standard/Password	clear	clear	clear	n/a	
Authentication	clear	clear	encrypted	MAC	
Encryption	clear	encrypted	encrypted	MAC	

 MAC Value used for data integrity check is a checksum encrypted by a session key

Early Findings

A simple replay attack isn't possible...

And we have no expertise or time for cryptoanalysis of authentication procedure either (6 weeks!!)

What's left?

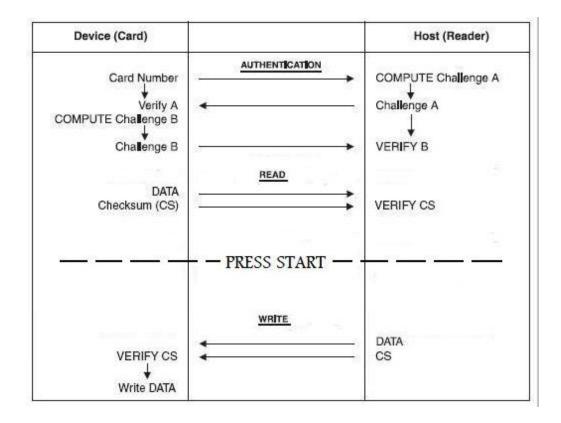
A Hypothesis...

What would be the most logical implementation of this type of system?



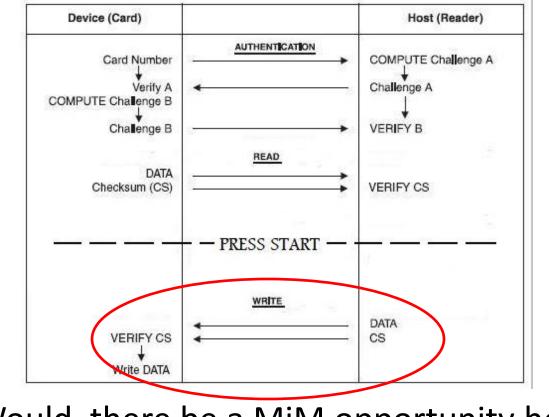
A Hypothesis...

What would be the most logical implementation of this type of system?



A Hypothesis...

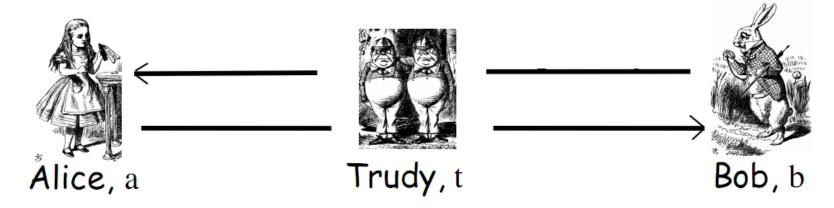
What would be the most logical implementation of this type of system?



Would there be a MiM opportunity here?

Alice, Bob and Trudy...

Man in the middle concept



Can we apply this in hardware?

MACHINE-TO-CARD COMMUNICATION

Sniffing Tools:

Modified Smart Card

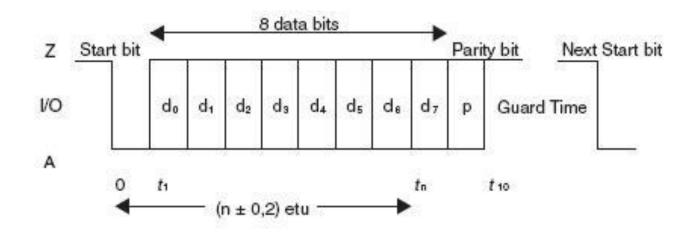


Sniffing Tools:

- Modified Smart Card
- Logic Analyzer



- 9600 Baud UART
- Byte-By-Byte data transmission in the form:



Groups of Bytes make up Commands

Communication Flow (Before Pressing Start):

• Card Sends Answer To Reset (ATR)

- Card Sends Answer To Reset (ATR)
- Machine Reads 8-Byte Cryptogram and 7-Byte ID

 Machine uses Cryptogram to compute challenge value

- Card Sends Answer To Reset (ATR)
- Machine Reads 8-Byte Cryptogram and 7-Byte ID

 Machine uses Cryptogram to compute challenge value
- Machine Sends Authentication Request Command

 Machine sends 16-Byte challenge value and nonce
 Card uses challenge to compute new cryptogram

- Card Sends Answer To Reset (ATR)
- Machine Reads 8-Byte Cryptogram and 7-Byte ID

 Machine uses Cryptogram to compute challenge value
- Machine Sends Authentication Request Command

 Machine sends 16-Byte challenge value and nonce
 Card uses challenge to compute new cryptogram
- Machine Reads *new* 8-Byte Cryptogram
 New cryptogram is verified

- Card Sends Answer To Reset (ATR)
- Machine Reads 8-Byte Cryptogram and 7-Byte ID

 Machine uses Cryptogram to compute challenge value
- Machine Sends Authentication Request Command

 Machine sends 16-Byte challenge value and nonce
 Card uses challenge to compute new cryptogram
- Machine Reads *new* 8-Byte Cryptogram
 New cryptogram is verified
- Machine Reads from User Zones

Communication Flow (After Pressing Start):

• Machine Reads from User Zones

Communication Flow (After Pressing Start):

• Machine Reads from User Zones

 Machine Writes Newly Decremented Balance to User Zones (Using 3 Write Commands)

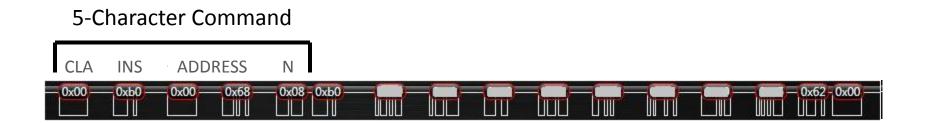
Communication Flow (After Pressing Start):

• Machine Reads from User Zones

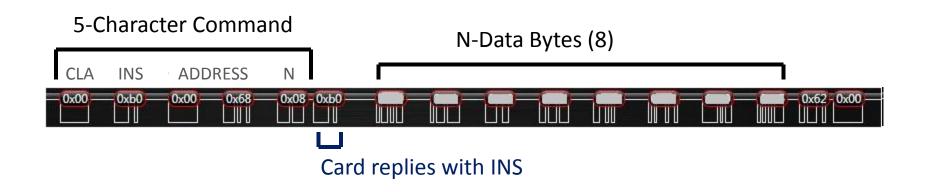
 Machine Writes Newly Decremented Balance to User Zones (Using 3 Write Commands)

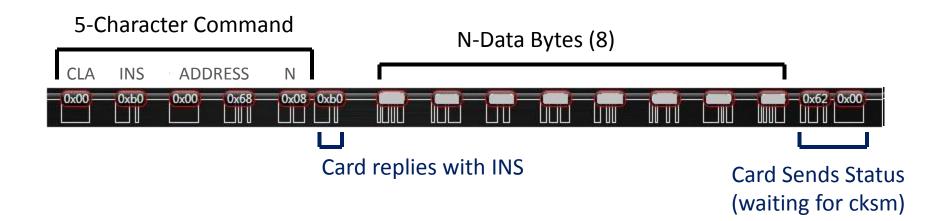
** Read and Write Commands Require Trailing CheckSum Cmds

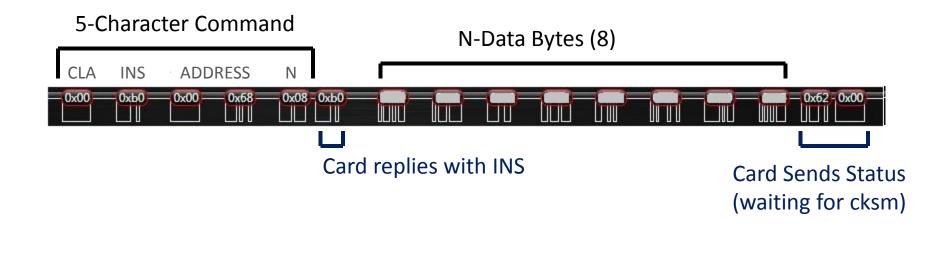




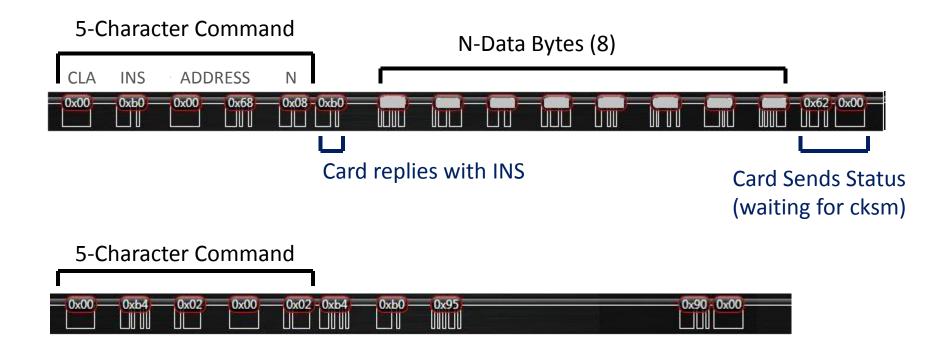


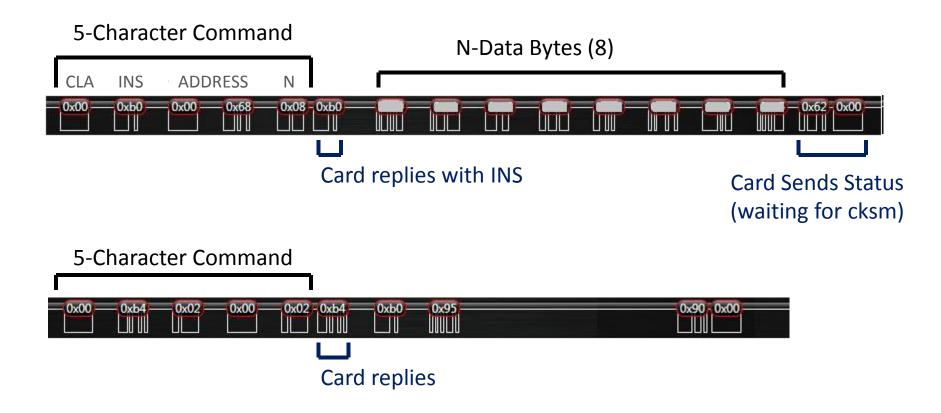


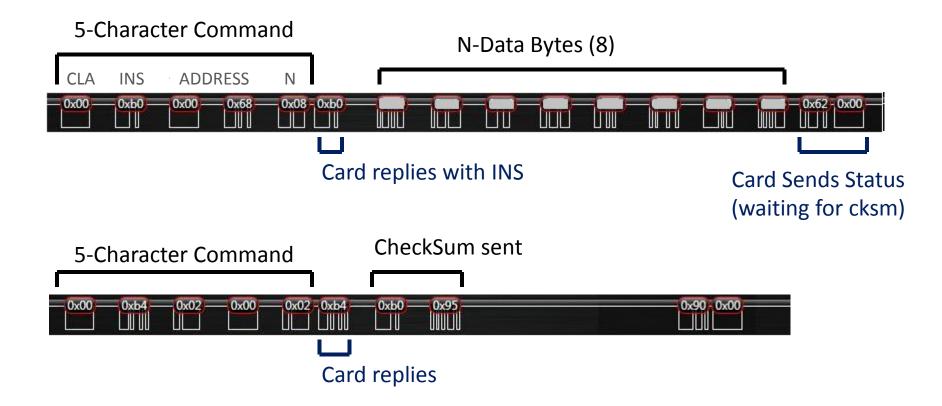


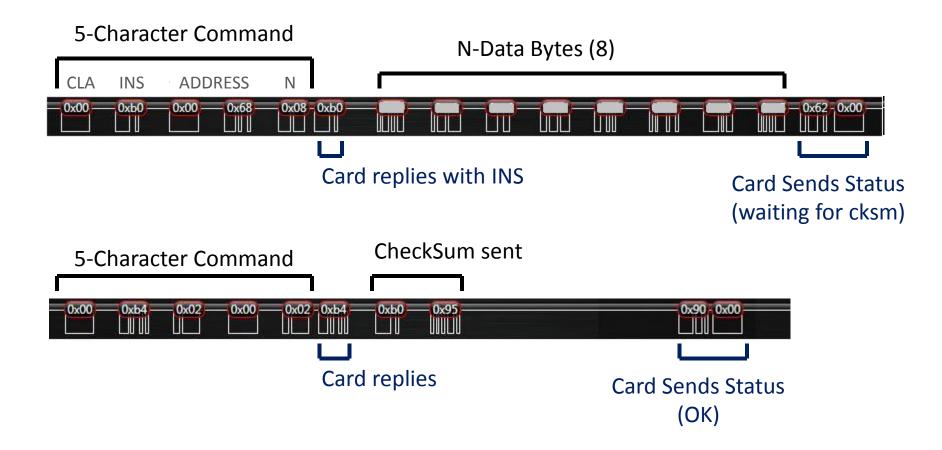


0x00	0xb4	0x02	0x00	0x020xb4	0xb0	0x95	0x90 - 0x00

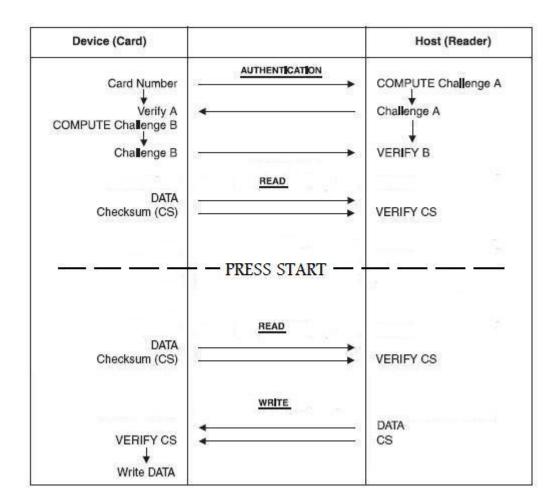




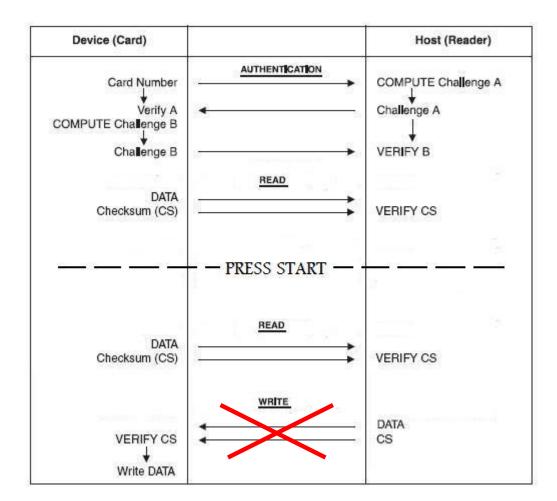




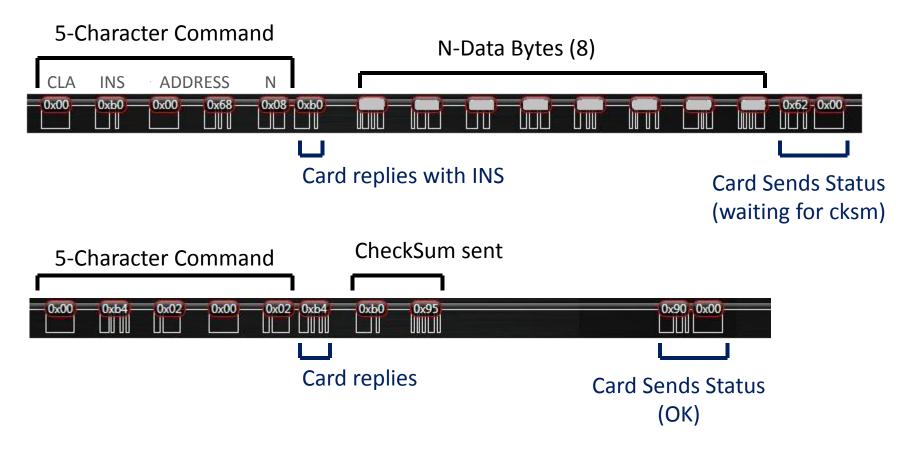
To Recap:



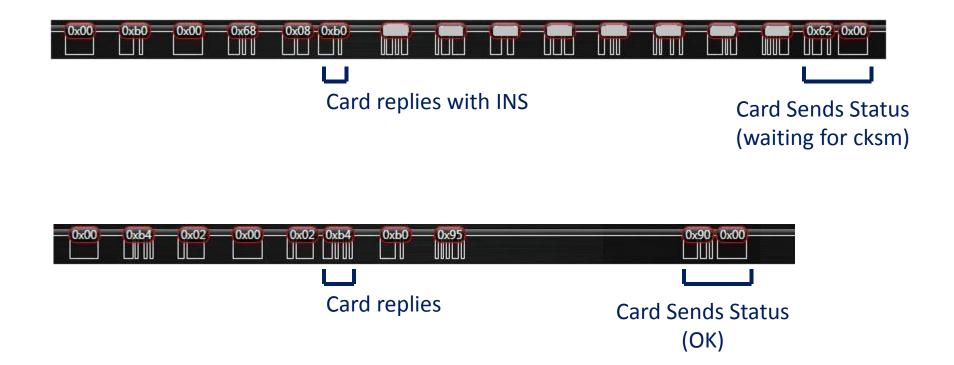
The Plan:



Recall the writing process...



... the Machine expects non-unique replies



THERE IS NO SECURE RECEIPT AFTER WRITING!

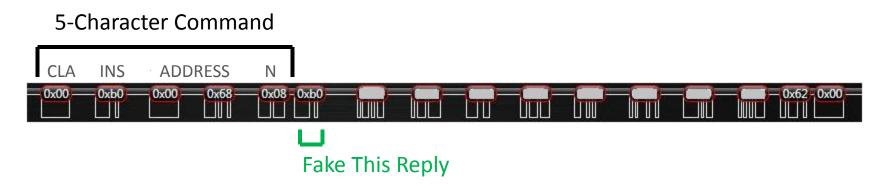
- The card confirms that data from the Host is legitimate
- However, the Host is not assured that the writing actually took place

What a hacker needs to do:

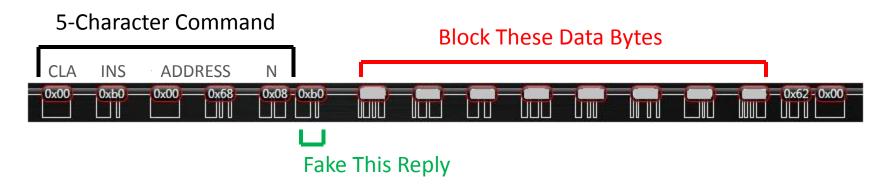
5-Character Command



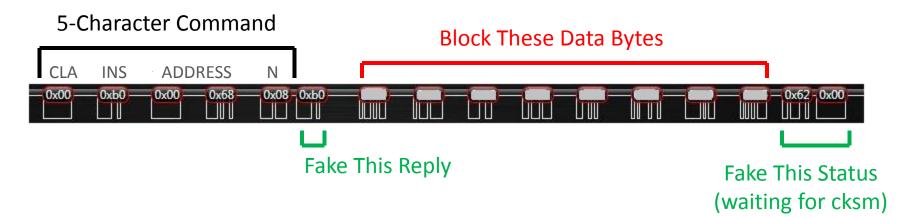
What a hacker needs to do:

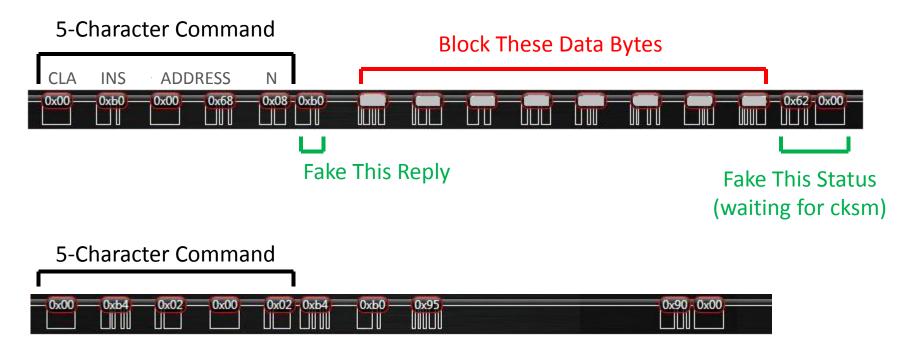


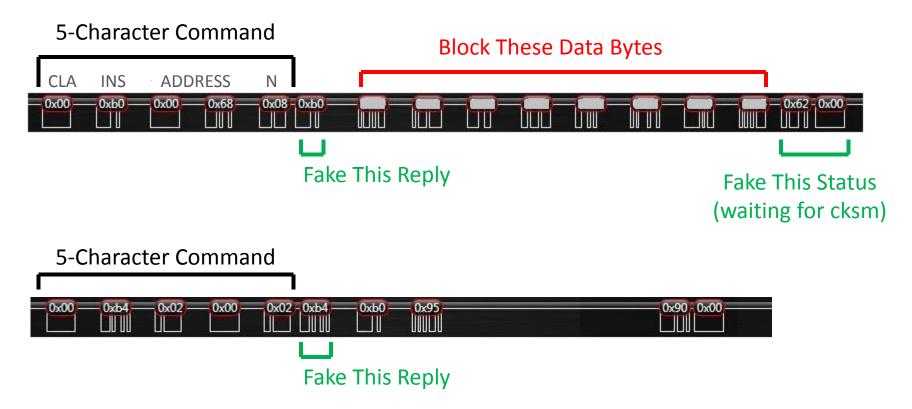
What a hacker needs to do:

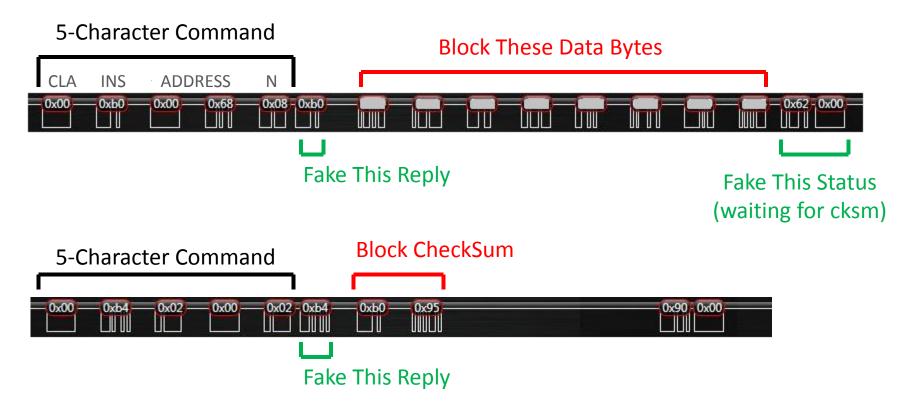


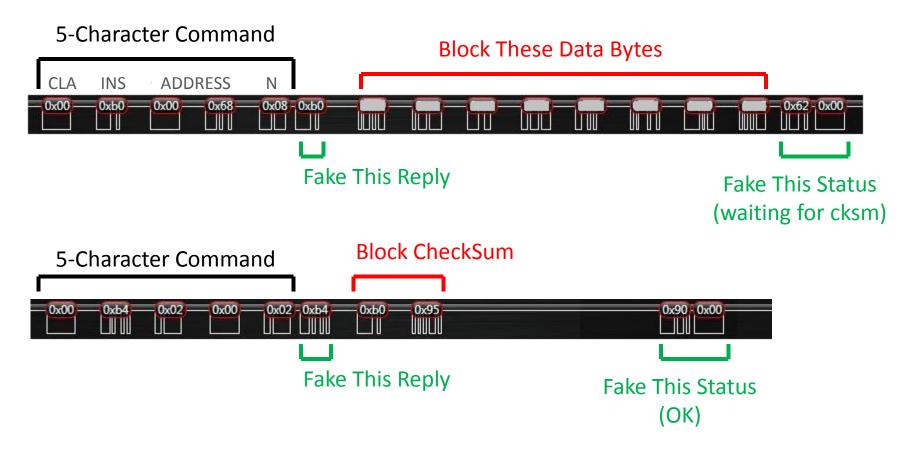
What a hacker needs to do:

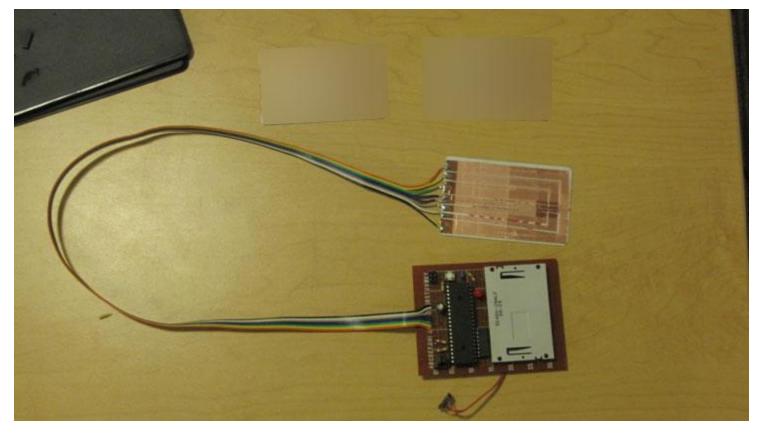




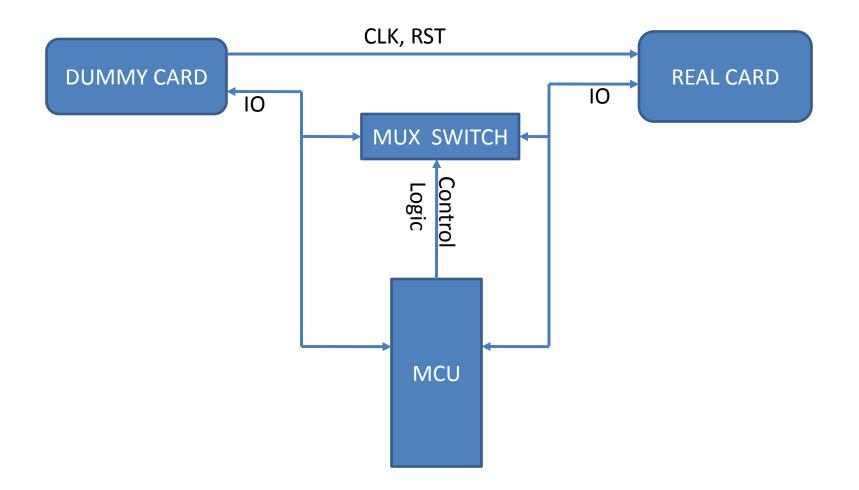


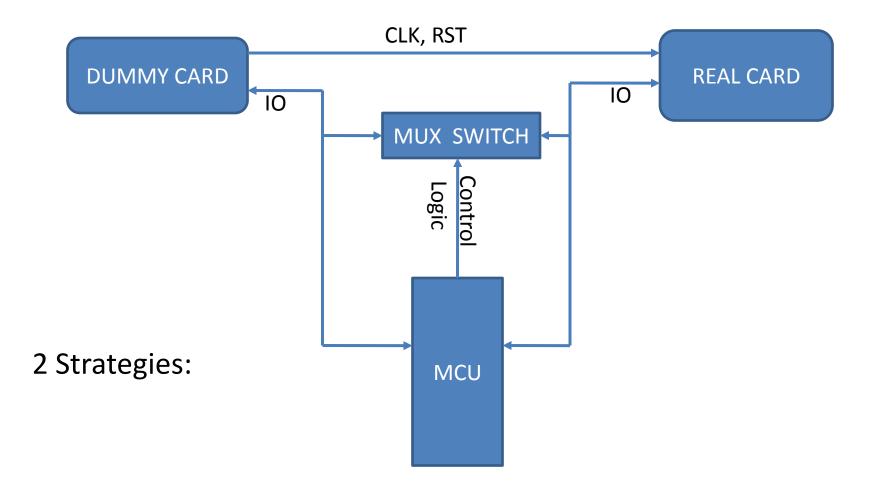


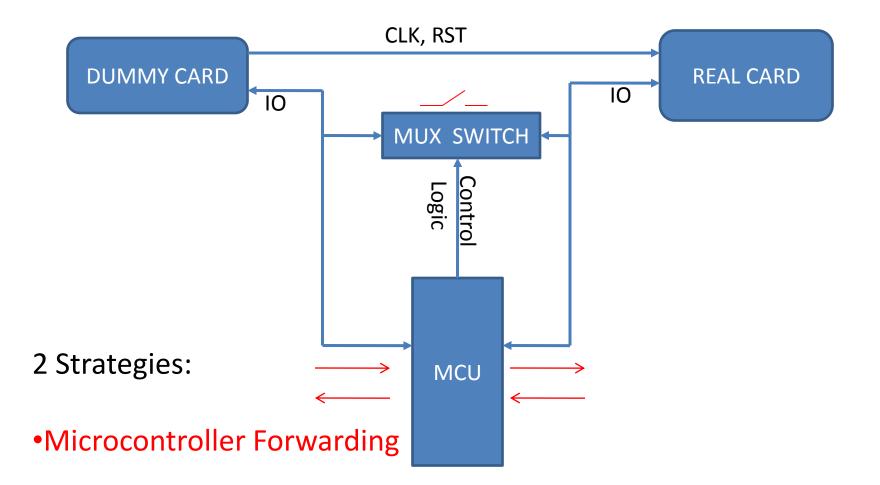


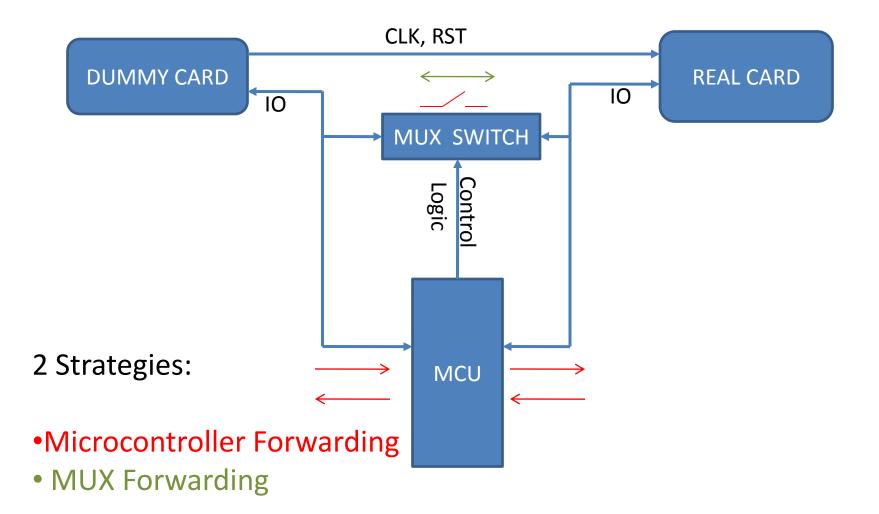


VULNERABILITY EXPLOITATION









10 M Samples	@ 1 MHz	Start Simulation	
		0ms +10ms	
1 Async Serial		0x3b-0xb2-0x11-0x00-0x10-0x80-0x00-0x04	
2 Async Serial		0x3b-0xb2-0x11-0x00-0x10-0x80-0x00-0x04	
3 Async Serial	01		

- LINE 1 : Machine to Microcontroller Node
- LINE 2 : Microcontroller to Card Node

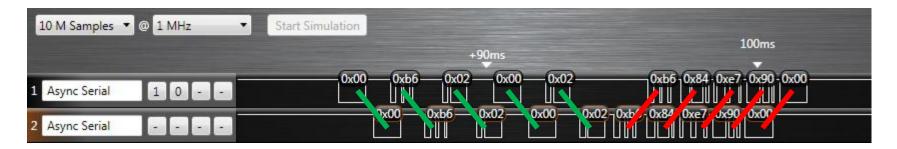
Answer To Reset is being forwarded through the MUX switch (No Delay)

10 M Samples	• @ 1 MHz •	Start Simulation	
		Oms	+10ms
1 Async Serial			0-0x10-0x80-0x00-0x04
2 Async Serial		0x3b-0xb2-0x11-0x00	0-0x10-0x80-0x00-0x04
3 Async Serial	01		

10 M Samples	• @ 1 MHz •	Start Simulation			
			+9 <u>0</u> ms		100ms
1 Async Serial	10	0x00	-0xb60x020x00	-0x020xb6	0x84 · 0xe7 · 0x90 - 0x00
2 Async Serial		0x	and a second sec		- 0xe7 - 0x90 - 0x00

Subsequent commands are being forwarded through the microcontroller

10 M Samples	• @ 1 MHz •	Start Simulation	
		Oms	+10ms
1 Async Serial			0-0x10-0x80-0x00-0x04
2 Async Serial		0x3b-0xb2-0x11-0x00	0-0x10-0x80-0x00-0x04
3 Async Serial	01		



Subsequent commands are being forwarded through the microcontroller (1 Character Delay)









Recognized Write Command



Blocked all further communication to the card (Open Mux Switch)



Recognized Write Command



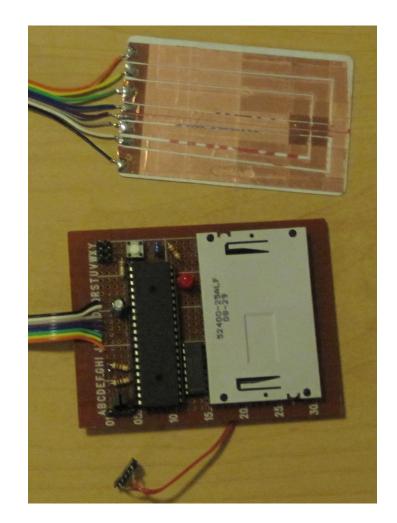
Blocked all further communication to the card (Open Mux Switch)

Forged Confirmation of Write Back to Machine

The Final Product:

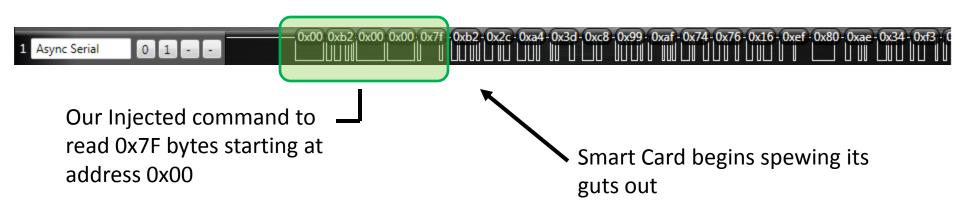
Value of Components:

Less than \$15



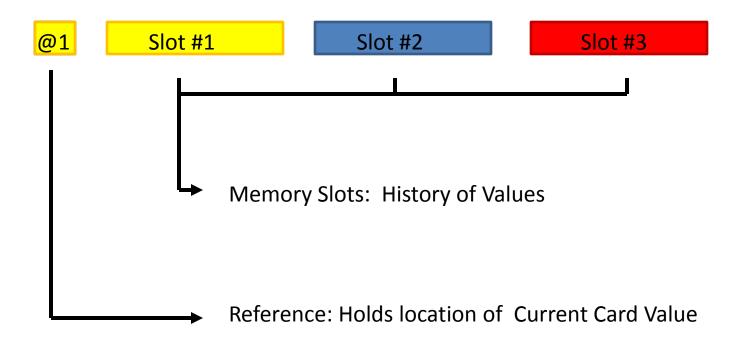
- Session Hijacking:
 - Allow system to reach the authenticated state
 - MIM communicates exclusively with smart card
 - Dump the protected user zones

- Session Hijacking:
 - Allow system to reach the authenticated state
 - MIM communicates exclusively with smart card
 - Dump the protected user zones

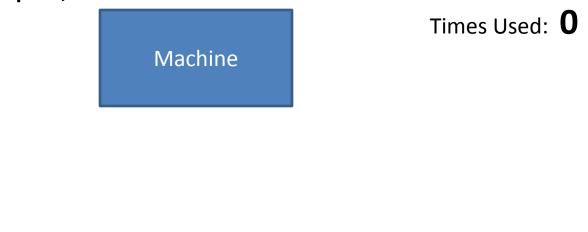


• Analysing the Protected User Memory:

- Analysing the Protected User Memory:
 - System uses History Buffer

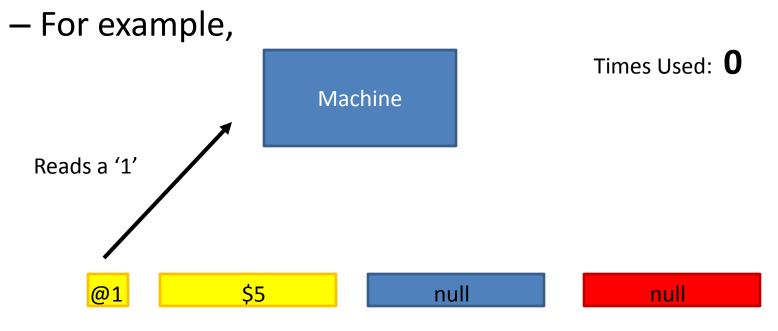


- Analysing the user memory:
 - System uses History Buffer
 - For example,

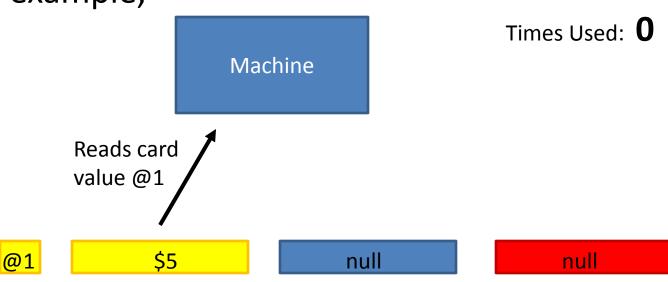




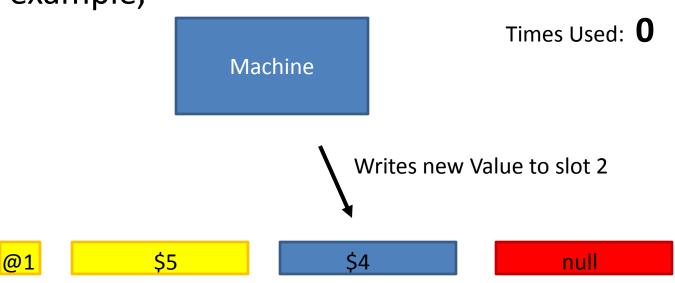
- Analysing the user memory:
 - System uses History Buffer



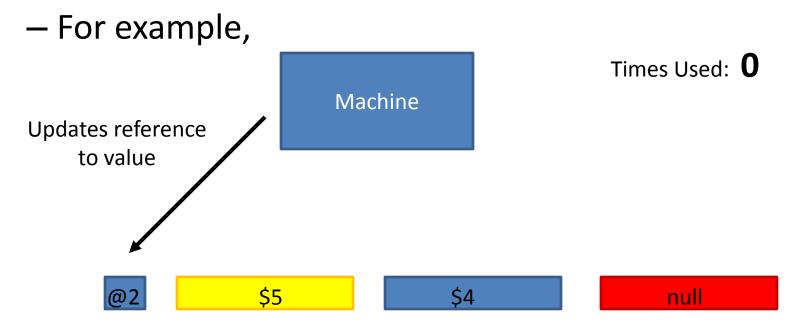
- Analysing the user memory:
 - System uses History Buffer
 - For example,



- Analysing the user memory:
 - System uses History Buffer
 - For example,



- Analysing the user memory:
 - System uses History Buffer

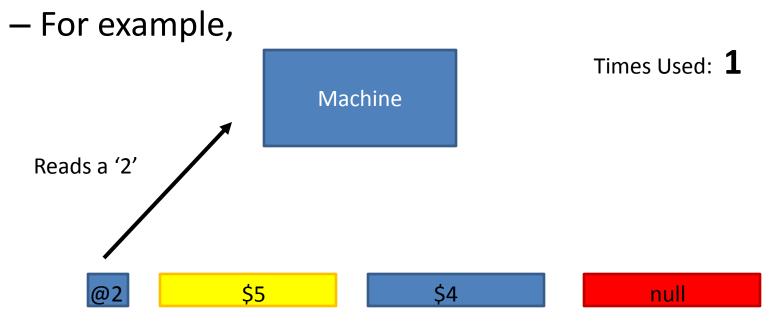


- Analysing the user memory:
 - System uses History Buffer
 - For example,

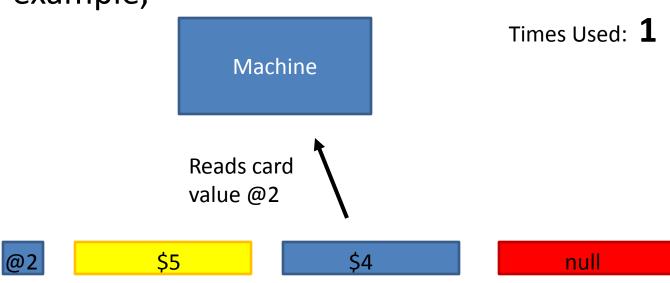




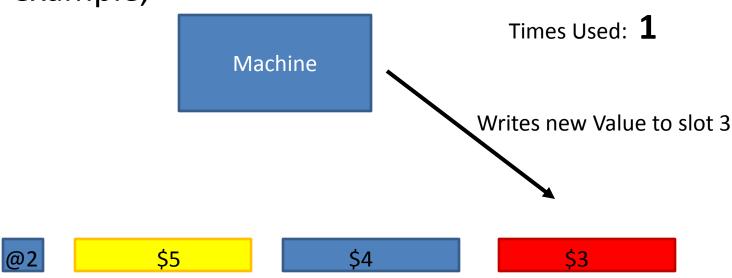
- Analysing the user memory:
 - System uses History Buffer



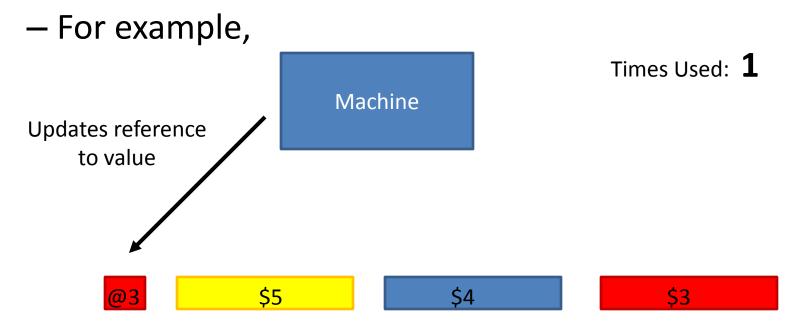
- Analysing the user memory:
 - System uses History Buffer
 - For example,



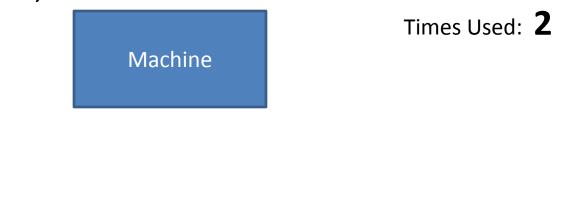
- Analysing the user memory:
 - System uses History Buffer
 - For example,



- Analysing the user memory:
 - System uses History Buffer

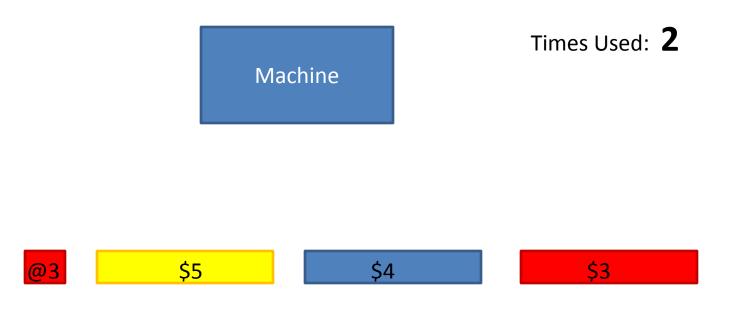


- Analysing the user memory:
 - System uses History Buffer
 - For example,

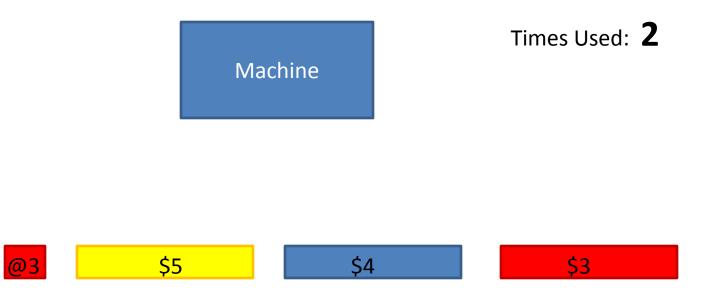




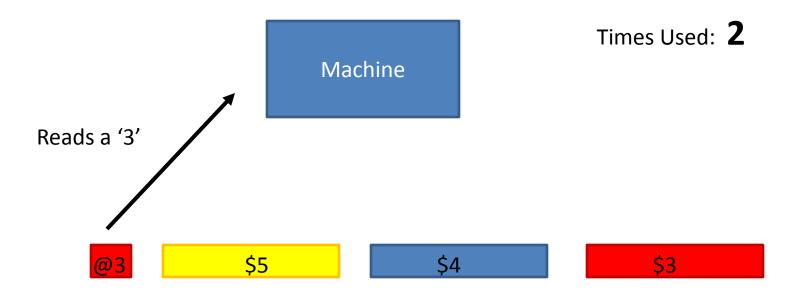
• Can this system be manipulated?



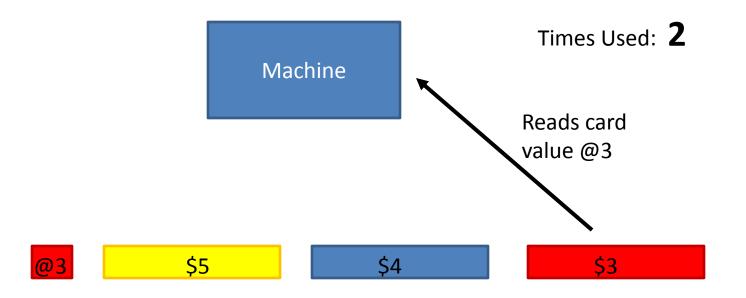
- Can this system be manipulated?
 - When the final memory slot is reached, what if...



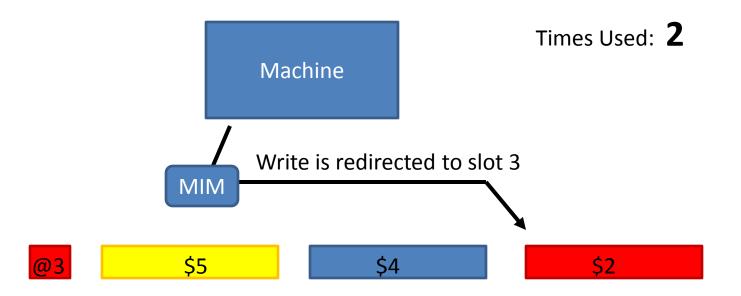
- Can this system be manipulated?
 - When the final memory slot is reached, what if...



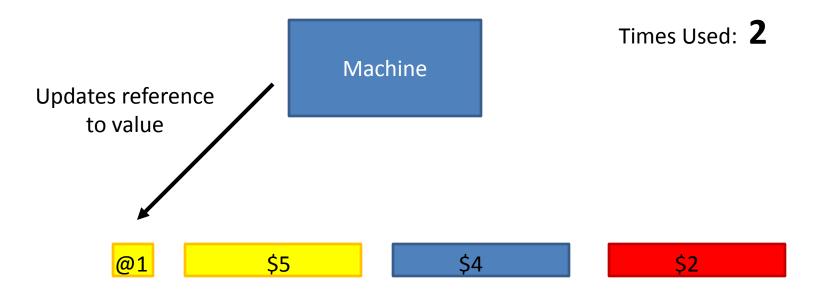
- Can this system be manipulated?
 - When the final memory slot is reached, what if...



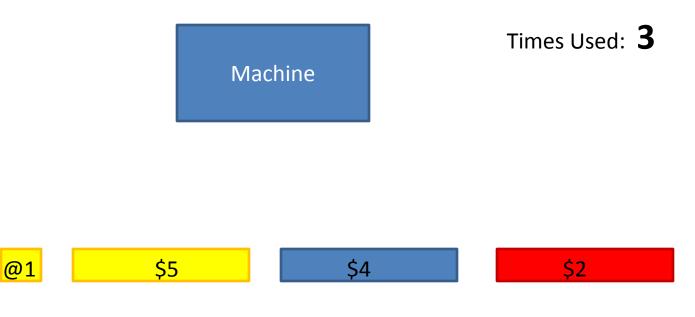
- Can this system be manipulated?
 - When the final memory slot is reached, what if...



- Can this system be manipulated?
 - When the final memory slot is reached, what if...



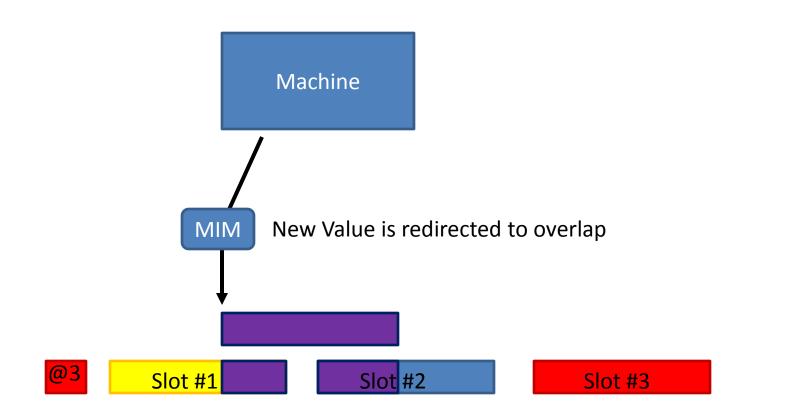
- Can this system be manipulated?
 - When the final memory slot is reached, what if...



The card will now hold the original \$5 value!

• This attack is possible if we are able to redirect the write command

- What else is possible with a redirected write?
 - Change the value maybe...?



What value will slot #1 hold?

• So we tried to redirect the write...

But, were denied. 🛞





Summary

- Communications that end in write command are not secure!
 - Even though Atmel App note recommends this exact behaviour



- Fix:
 - End Communication with a read command to verify the new card value

Summary

- Dump User Zone memory using MiM and session Hijacking
 - Fix:
 - Use encryption mode
 - Host side tamper detection(voltage/current monitor)

MONTAGE VIDEO TIME!!!