RSA CONFERENCE 2014

FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Share.
Learn.
Secure.

Capitalizing on
Collective Intelligence

‘2nd-Wave’ Advanced Threats: Preparing for Tomorrow’s Sophisticated Attacks

SESSION ID: ANF-W01

Nikos Triandopoulos

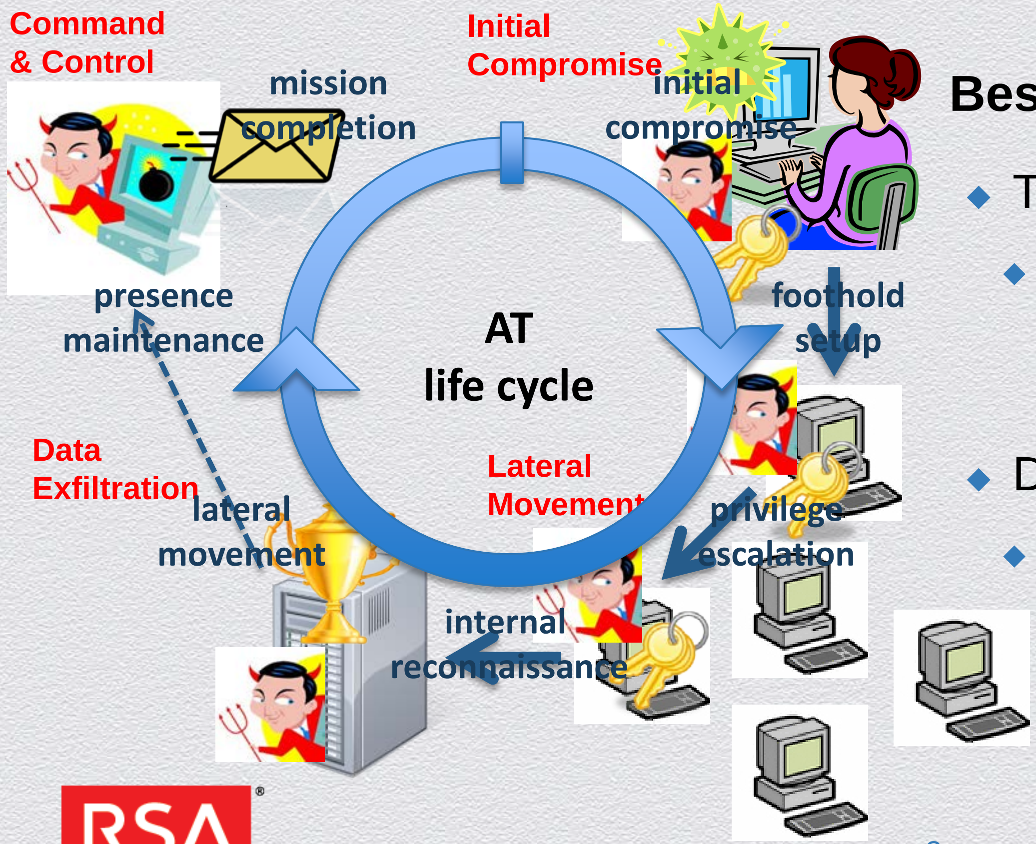
Principal Research Scientist
RSA Laboratories
RSA, The Security Division of EMC

Advanced Threats: Enterprises' toughest enemy

- Advanced Threats (ATs) are a serious risk facing enterprises today
  - comprise well-targeted, persistent attacks
  - aim at unauthorized data manipulation or exfiltration
  - employ rich attack vectors and unknown strategies
    - social engineering
    - zero-day malwares / vulnerabilities
    - low-and-slow progression

➔ Extremely hard-to-defend, often even hard-to-detect

The “canonical” attack cycle



Best defenses in security industry

- Tighter preventative practices
  - raise the protection fence
    - e.g., multi-factor authentication, data protections, access control, etc.
- Detection & forensics tools
  - visibility – analysis – action
    - e.g., security information event management (SIEM) systems, security analytics

‘2nd-Wave’ Advanced Threats

- Tougher, evolving adversaries who
  - grow in sophistication to become context aware and target specific
    - know “*what they attack and how it is protected*”
  - shift towards qualitatively stronger attack strategies

Achieve their objective <u>while trying to evade</u> defensive tools	→	**Achieve their objective <u>by first disarming</u> defensive tools**
(past / current)		(current / future)

In practice this means…

- If strong authentication is used, the attacker will steal
  - stored keys to clone authenticators
  - passwords to impersonate users
  - credentials to forge signatures
- If security logs are collected and analyzed, the attacker will
  - block the stream of reported logs
  - employ log-scrubbing malware to cover its tracks
  - tamper with host-side log generation software

This presentation

- Gain awareness of new type of threats
  - Examples of ‘2nd-wave’ ATs against current security practices
- Describe new solution concepts
  1. Anti-cloning enhancements for authentication devices
  2. Intrusion-resilient passcode/password verification
  3. Anti-breach hardening of SIEM systems
- Learn general strategies
  - How to harden security solutions to resist partial compromises

Contributors:
Kevin Bowers
John Brainard
Marten van Dijk
Catherine Hart
Ari Juels
Ron Rivest
Emily Shen
NT

RSACONFERENCE2014

FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Anti-cloning enhancements for authentication devices

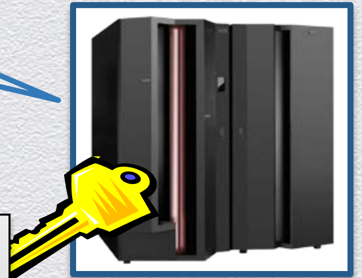
Problem: Cloning of authentication devices

- Theft of cryptographic key permits device (and user) impersonation!

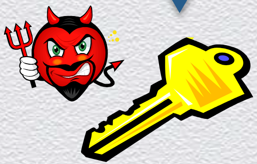
Alice's device

...

Hello, **Alice!**



key leakage



any authentication device:
one-time passcode hardware tokens,
software authenticators,
secret-key authentication schemes in sensors,
embedded devices or mobile phones, etc.

…ossible to distinguish!

…ly on key

2. clone is digitally identical to original

RSA

#RSAC

RSACONFERENCE2014

Key leakage is possible in many ways

- Device
  - side-channel attacks
  - physical tampering
  - key-extracting malware
- Authentication server
  - server compromise
- Key stores
  - data exfiltration of key records

Running example: One-time authentication tokens

- Representative case: resource-constraint authentication device

Hello, Alice!

Alice's device

"159 759" (+ PIN)

"159 759" (+ PIN)

- token cloning (& PIN phishing), user impersonation
- **no assumptions** on stored internal secret state or used passcode-generation method!

Solution: Use covert channel to signal token cloning

Key idea: Augment cryptographic key to allow detection of cloning attack

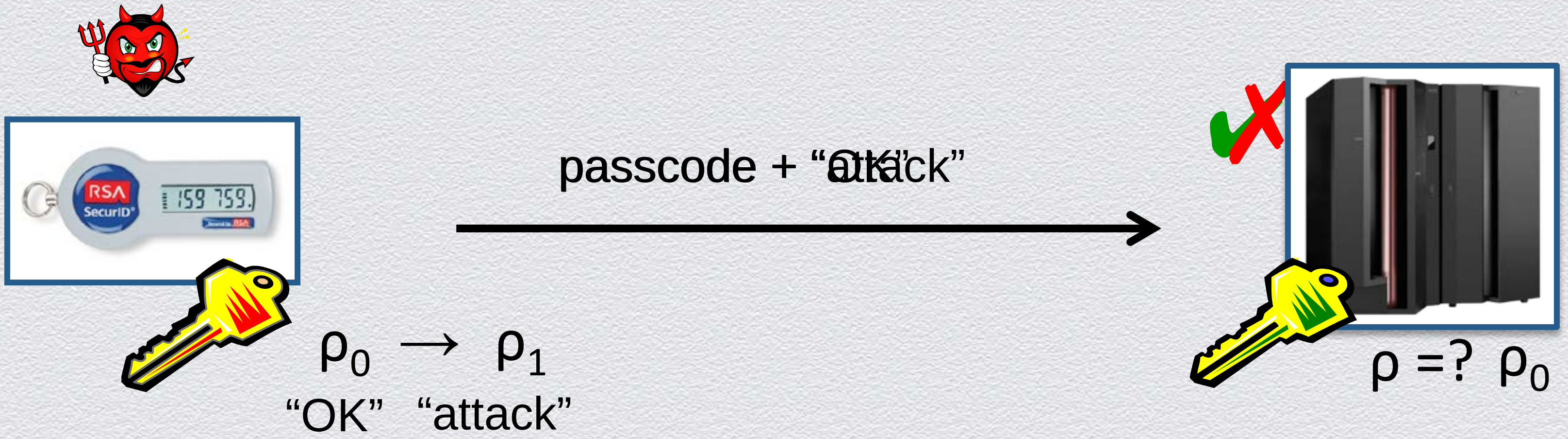
Token generates: passcode + signal

Signal type	Cloning detection		Assumption
1. Silent Alarm	token tampering or compromise	immediate	sensing capability at the token
2. Drifting Key	any leakage	eventual	regular token usage

key-based status update, secretly embedded into passcode

RSA®

Silent alarms

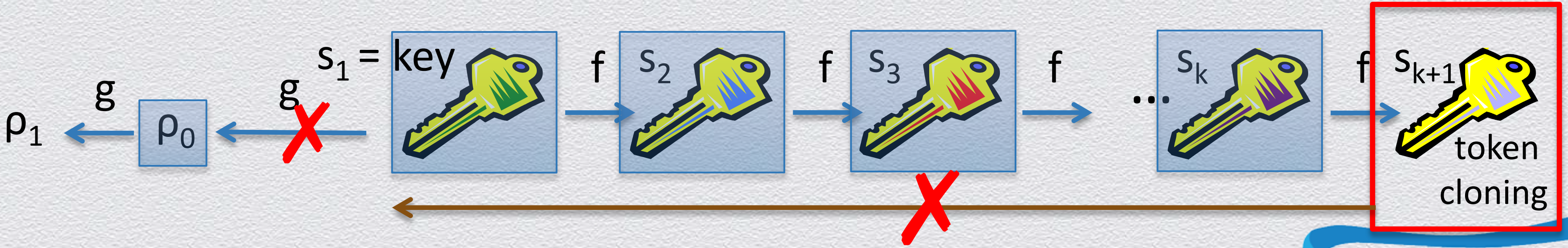


- Embed random secret “***health***” state $\rho_0 \in \{0, 1\}^n$ known by server
- Upon sensing tampering, change to random state $\rho_1 \in \{0, 1\}^n - \{\rho_0\}$
- Security parameter n controls signal secrecy

Secret and forward-secure state transitions

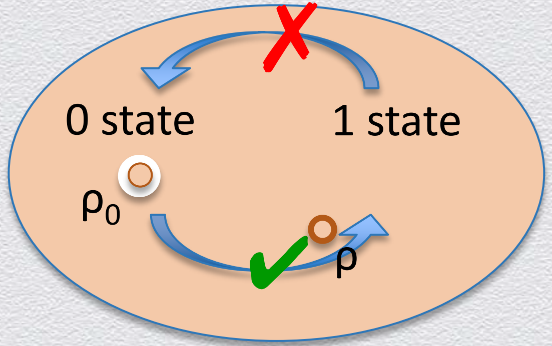
Health state transition from ρ_0 to ρ_1 should be

1. unpredictable: Attacker can't reset state to "OK"
   - e.g., derive pseudorandom states from key via one-way hashing
2. forward secure: Attacker can't learn "attack" state via a replay attack
   - e.g., update key irreversibly through one-way hashing

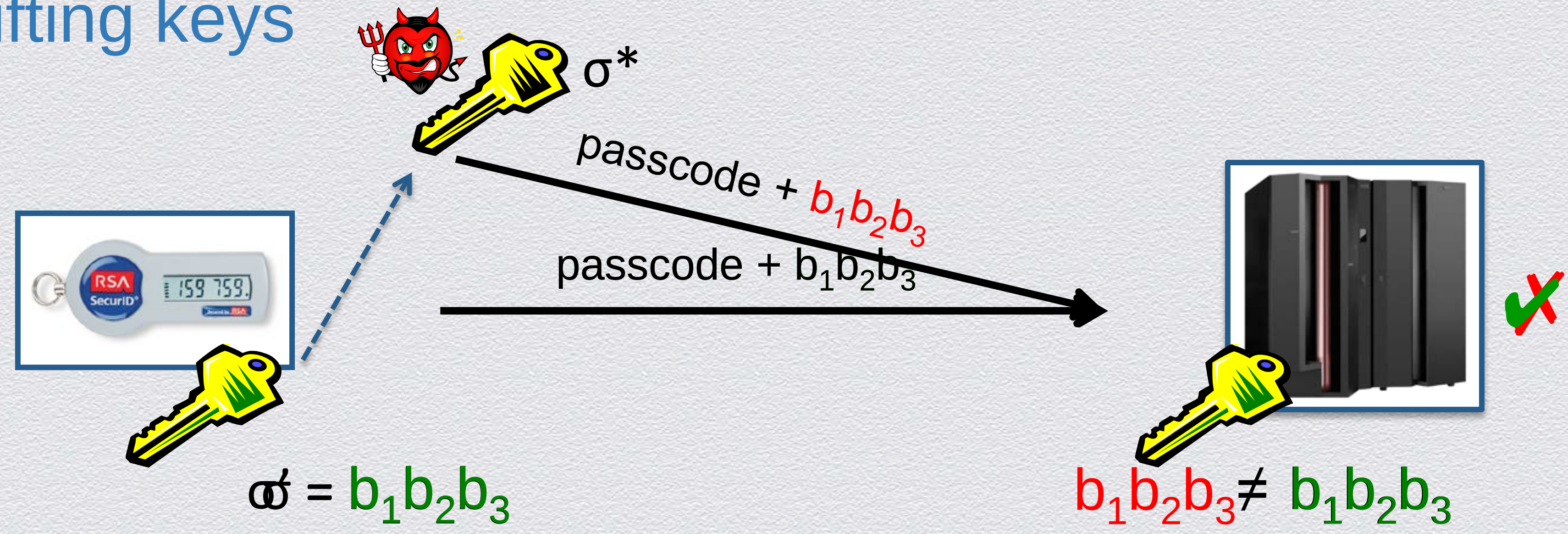


Properties of silent alarms

- Implements simple authenticated-encryption scheme on 1-bit alerts
- Biased authenticity
  - an adversary can only compute a “1” encoding, but not a “0” one
  - alarm is unchangeable, i.e., cannot be turned off, thus <u>persistent</u>
- One-time pad confidentiality
  - with secret ρ_0, an adversary cannot determine whether state ρ is a “0” or “1” encoding
  - alarm is undetectable, thus <u>silent</u>



Drifting keys



- Embed randomly and periodically evolving secret "***uniqueness***" state $\sigma \in \{0, 1\}^m$
- A cloned token's state σ^* will likely divert from σ
- Inconsistent states collected in parallel are eventually detected by server

#RSAC

RSA CONFERENCE 2014

Evolving drifting keys

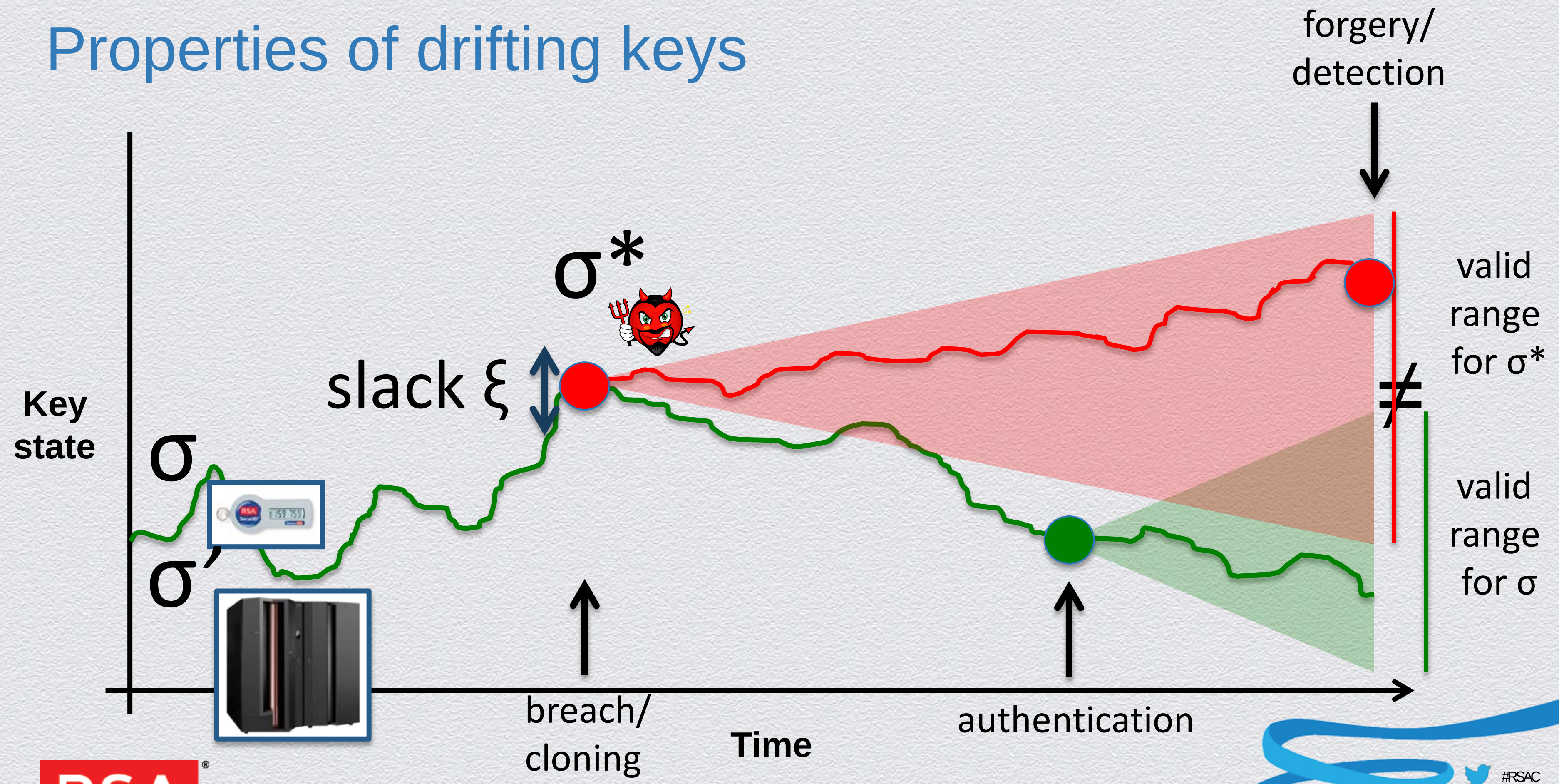
- Uniqueness state consists of 1-bit keys that “drift” regularly & randomly

$$\sigma = b_1b_2\ldots b_m \;\rightarrow\; \sigma' = b_1'b_2'\ldots b_m' \;\rightarrow\; \sigma'' = b_1''b_2''\ldots b_m'' \;\rightarrow\; \ldots$$

- Uniformly staggered updates
  - periodic round-robin bit(s) randomization
  - e.g., keep 7 bits and randomly update one bit every day

Mon	Tue	Wed	Thu	Fri	Sat	Sun
0	0	0	1	0	1	0

Properties of drifting keys



Transmitting health and uniqueness states

- New challenge: The token-to-server channel is very restricted
  - low-bandwidth: only available channel is embedding into passcode itself
    - each bit allocated to signal weakens the security of passcode
  - susceptible to human-transcription errors
    - signal should not be distorted due to passcode mistyping!
  - lossy: displayed passcodes are rarely typed in
    - e.g., >99.994% of 1-min passcodes are not typed in for 6 logins/week
- Solution: Compress each state down to 1 bit, then encode 2 bits into an "offset" that is added to the passcode

Signal compression, encoding and processing

Passcode generation (time t)

- State compression and encoding
  - derive pseudorandom masks x_t, y_t from current key s_t, $|x_t|=|\rho_t|$, $|y_t|=|\sigma_t|$
  - sample silent alarm bit $\mathbf{sa_t = \rho_t \bullet x_t}$
  - sample drifting-keys bit $\mathbf{dk_t = \sigma_t \bullet y_t}$
  - set offset C as secret encoding of $sa_t dk_t$
  - produce enhanced passcode $P_t \oplus C$ (using digit-wise mod 10 addition)

0	33333333
0	
0	33337777
1	
1	77773333
0	

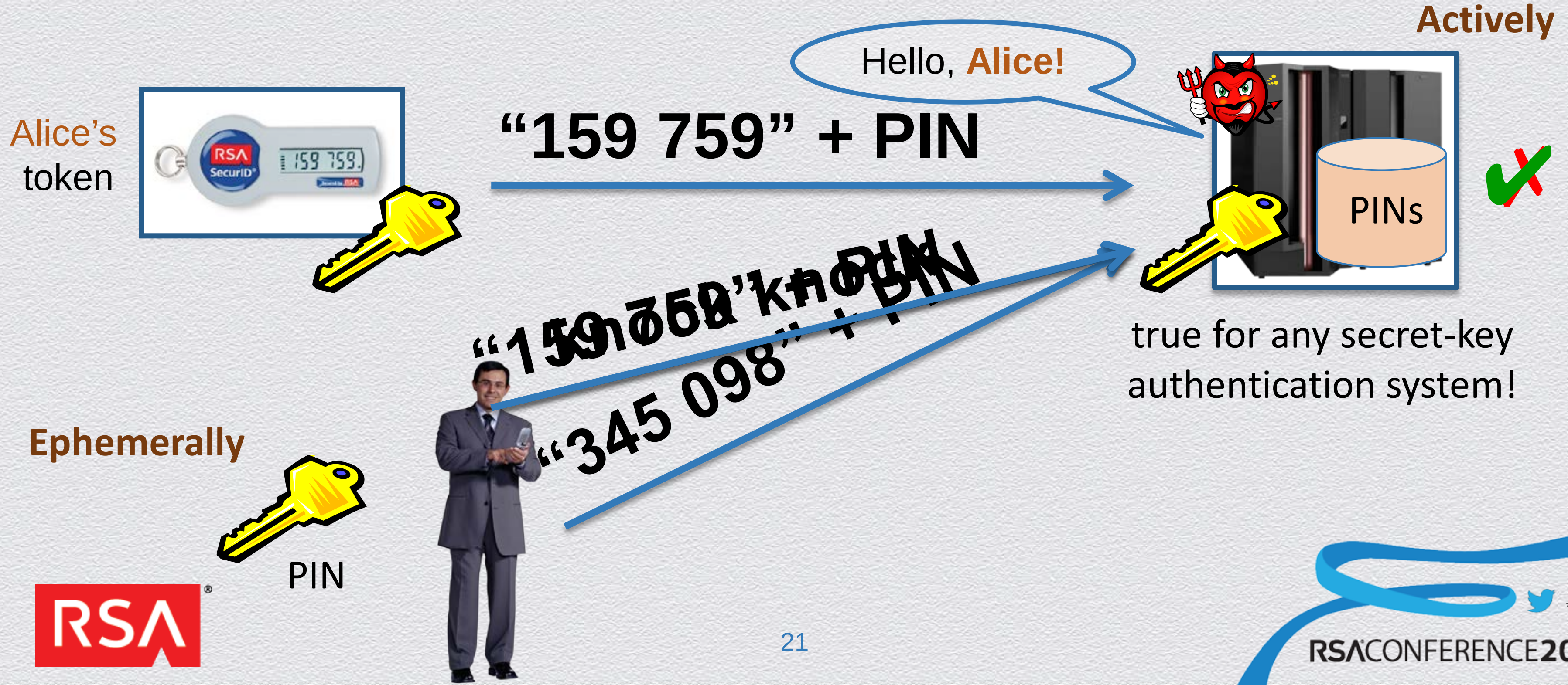
Passcode verification (time t)

- State recovery and checking
  - accept received passcode Q′ only if C = Q′-P_t is a valid codeword of secret code
  - decode C to recover sa_t and dk_t
  - perform probabilistic check $sa_t =? \rho_0 \bullet x_t$
    - perfect soundness, 50% false negative
    - 0.75 prob. of break-in detection in 2 logins
  - check for inconsistencies in set of equations $\{dk_t = \sigma_t \bullet y_t |$ login at t$\}$, i.e., if system becomes infeasible

Intrusion-resilient passcode/password verification

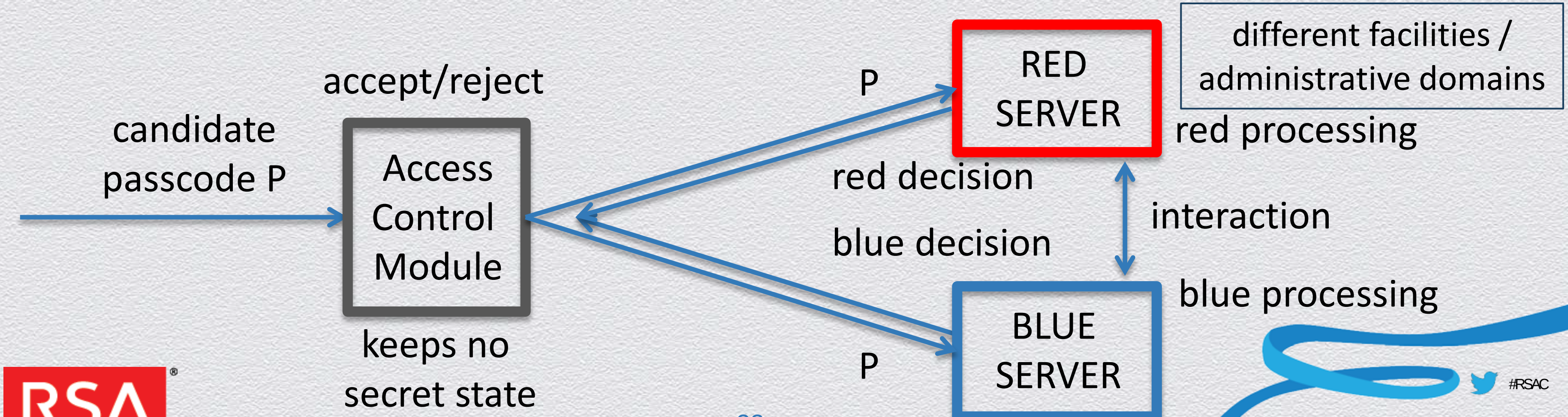
Problem: Compromise of authentication server

- Direct breach at authentication server is catastrophic!



Solution: Split-server verification

- Key idea: Distribute passcode/PIN verification across two servers
  - Red server verifies “half” the credentials; blue server verifies other “half”
  - Authentication decision relies on both outputs
    - Compromise of one server gives no/little advantage to attacker



Split-server passcode verification

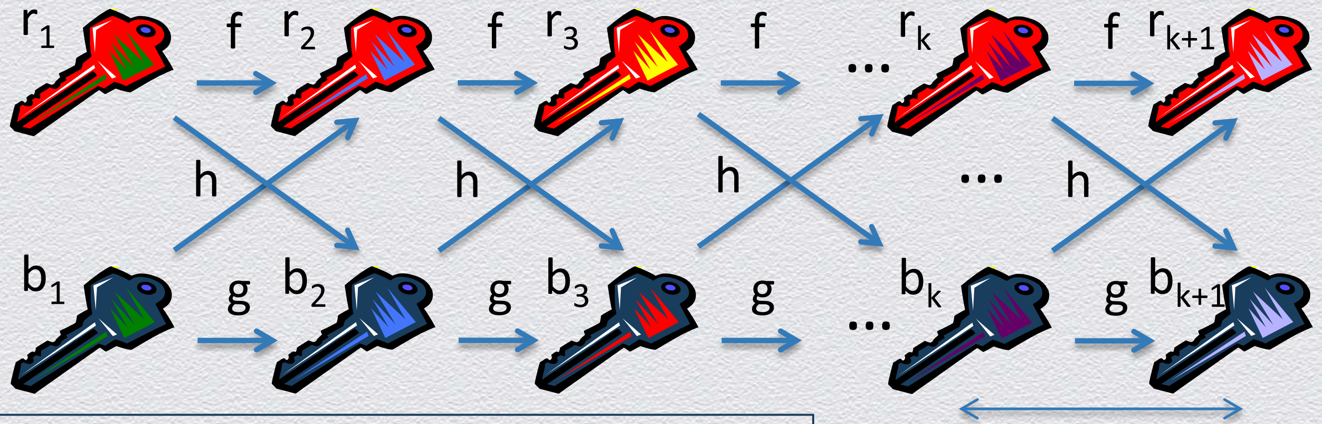
- Token-side: Employ two distinct (fixed or forward-secure) secrets
  - red secret r is used to derive red partial passcode P_R
  - blue secret b is used to derive red partial passcode P_B
  - final passcode P is sum $P_R \oplus P_B$ (digit-wise modulo 10)
- Server-side: Red/blue server returns local accept/reject decision; candidate passcode P′ is accepted if both servers locally accept
  - crypto approach: red and blue run privately equality test on $P'-P_R$, P_B
  - non-crypto approach: red sends least significant half of P_R to blue and verifies the most significant half of candidate passcode (and vice versa)

Protecting against double-server attacks

- Goal: defend against non-simultaneous breach of both blue and red servers
- Use forward-secure red/blue partial secrets that periodically “mix”

$r_2=f(r_1,h(b_1))$

$b_2=g(b_1,h(r_1))$

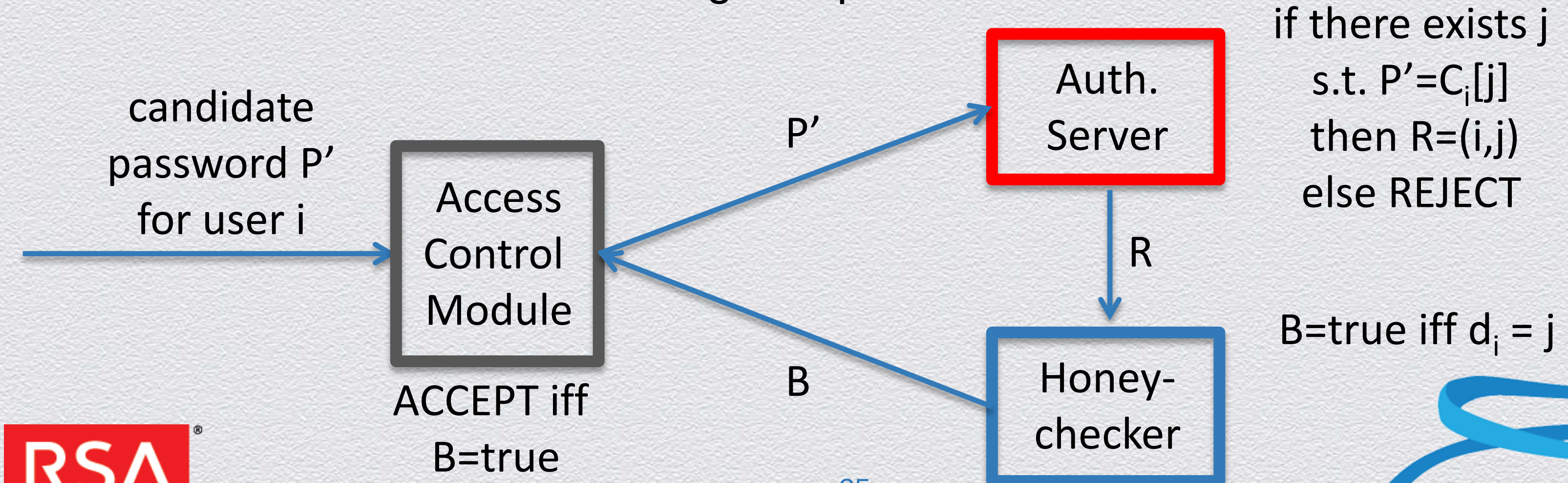


as long as servers are not both compromised in the same day the authentication system remains secure

e.g., 24 hours

Split-server password verification: Honeywords

- Based on decoy passwords, aka honeywords
  - Red stores user's i real password P_i and k-1 fake ones in unlabeled set C_i
  - Blue server stores the index d_i of P_i in set C_i
  - Password verification through sequential checks



RSA CONFERENCE 2014
FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Anti-breach hardening of SIEM systems

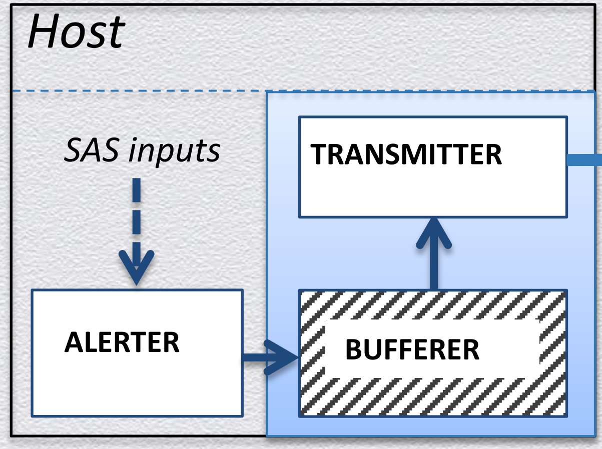
Problem: Secure chain of custody in security analytics

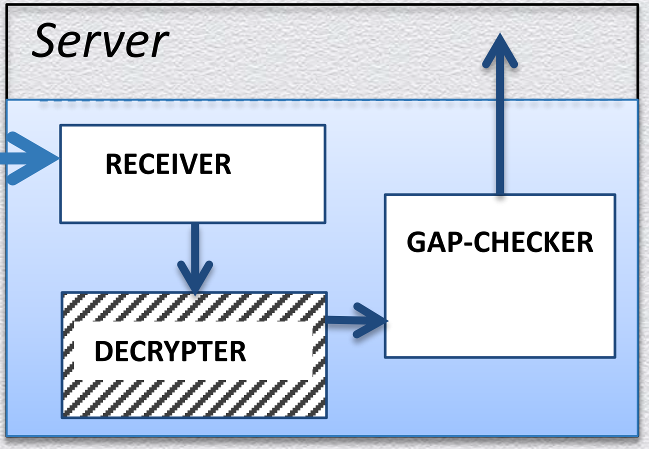
- Security alert systems often employ unreliable channels!
- Security alert systems constitute a direct target of a '2nd-wave' AT!
  - an attacker may discover, observe or read alert transmissions
    - …and accordingly adapt its attack strategy based on SAS behavior!
  - an attacker may tamper, suppress or block alert transmissions
    - …and eventually disrupt SAS functionality (e.g., using log-scrubbing malware)!

Solution: PillarBox, a secure alert-relaying tool

- ensures against alert suppression or tampering
- conceals alerting activity
- features self-protection, transmits alerts persistently
- is agnostic of the exact SAS in use

PillarBox architecture





TRANSMITTER-RECEIVER

schedule & execute crypto-enhanced host-to-server buffer transmissions

ALERTER

implements SAS, monitors host to identify events against a set of alert rules, creates alert messages and relays them to BUFFERER

BUFFERER-DECRYPTER

implement crypto-assisted reliable channel & report integrity failures

GAP-CHECKER

reconstructs alert stream, checks for missing alerts, reports “heartbeat” or “gap alert” failures

1. Buffering alerts

(FS) integrity ✓
(FS) confidentiality ✓

- As soon as they are generated, alerts are
  - **signed** and **encrypted** using a forward-secure secret key (shared by the server and host) and then stored in a buffer at the host
  - periodically or on demand (e.g., every t alerts) **transferred** to the server

2. Retransmitting alerts

(FS) integrity ✓

(FS) confidentiality ✓

persistence ✓

- As before, but now alerts
  - are **not deleted** from buffer but are **transferred redundantly**
    - e.g., when a new alert is generated all buffered a

persistence:
missing alerts can only be attributed to an attack, thus allowing to signal a “meta alert”

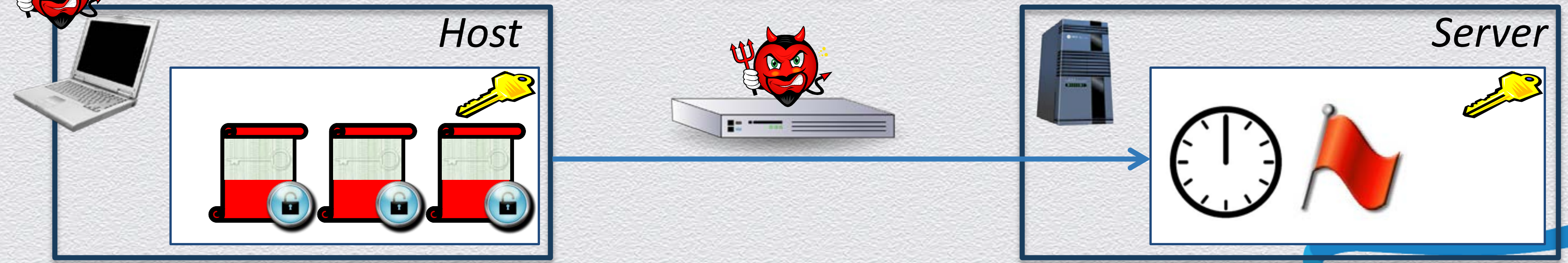
3. Checking heartbeat

(FS) integrity ✓
(FS) confidentiality ✓
persistence ✓
failure detection ✓
traffic concealment ✓

- As before, but now alerts
  - are transmitted **periodically** (in regular time intervals)
  - if failed to reach the server, they signal a **“heartbeat” failure** of SAS

failure detection:
imposes a minimum frequency of transmission
(allows an upper bound on successful detection)

traffic concealment:
imposes a regular pattern of transmissions
(so alerts can be de-correlated)

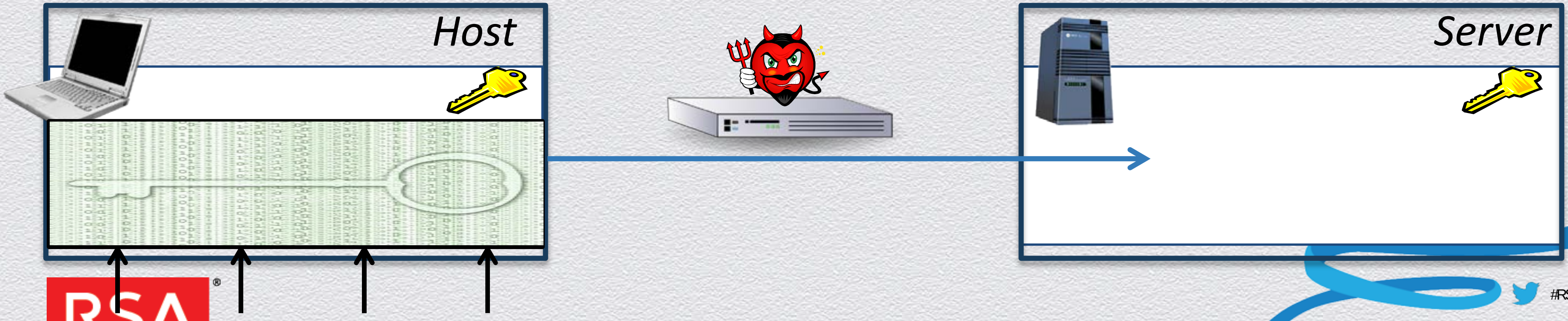


4. Encrypting fixed-size buffers

(FS) integrity ✓ **failure detection** ✓
(FS) confidentiality ✓ **traffic concealment** ✓
persistence ✓ **stealth** ✓

- As before, but now alerts
  - are stored in an initially **random, fixed-size** bu
  - are transmitted periodically **encrypted** as a wh
  - if failed to reach the server, they signal a **“gap**

stealth:
alerting mechanism is completely
hidden from attacker
(at some communication overhead)



Summary of solutions

Intrusion-resilient (two-factor) authentication

Intrusion-resilient security in log collection

- Key technologies
  - key rotation
  - covert channels
  - forward security
  - authenticated encryption
  - split-server verification
  - secure log buffering
  - …

References

- Drifting keys: Kevin D. Bowers, Ari Juels, Ronald L. Rivest, Emily Shen, “Drifting Keys: Impersonation Detection for Constrained Devices”, INFOCOM 2013: 1025-1033
- Split-server authentication: John Brainard, Ari Juels, Burt Kaliski, Michael Szydlo, “A New Two-server Approach for Authentication with Short Secrets”, USENIX Security 2003: 201-214
- Honeywords: Ari Juels, Ronald L. Rivest, “Honeywords: Making Password-cracking Detectable”, ACM CCS 2013: 145-160
- PillarBox: Kevin D. Bowers, Catherine Hart, Ari Juels, Nikos Triandopoulos, “Securing the Data in Big Data Security Analytics”, IAC ePrint Archive 2013: 625, http://eprint.iacr.org/2013/625

Thank you!

Nikos Triandopoulos

nikolaos.triandopoulos@rsa.com