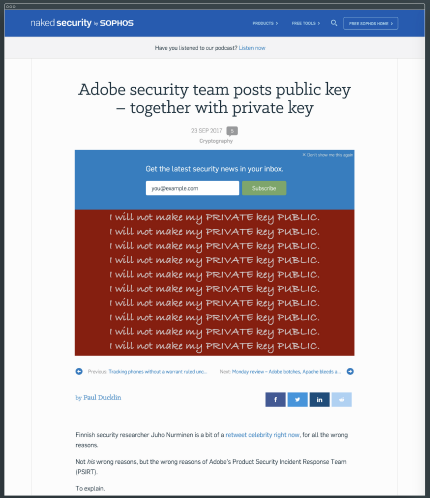
Revisiting Updatable Encryption: Controlled Forward Security, Constructions and a Puncturable Perspective

Daniel Slamanig (UniBw)* & Christoph Striecks (AIT)
Presentation given by Roman Langrehr (ETH Zurich)

TCC, November 30, 2023

*Work done while author was with AIT Austrian Institute of Technology



nakedsecurity by SOPHOS

PRODUCTS > FREE TOOLS > FREE SOPHOS HOME >

Have you listened to our podcast? Listen now

Adobe security team posts public key – together with private key

23 SEP 2017 · 5

Cryptography

Get the latest security news in your inbox.

you@example.com Subscribe

I will not make my PRIVATE key PUBLIC.
I will not make my PRIVATE key PUBLIC.
I will not make my PRIVATE key PUBLIC.
I will not make my PRIVATE key PUBLIC.
I will not make my PRIVATE key PUBLIC.
I will not make my PRIVATE key PUBLIC.
I will not make my PRIVATE key PUBLIC.
I will not make my PRIVATE key PUBLIC.
I will not make my PRIVATE key PUBLIC.
I will not make my PRIVATE key PUBLIC.

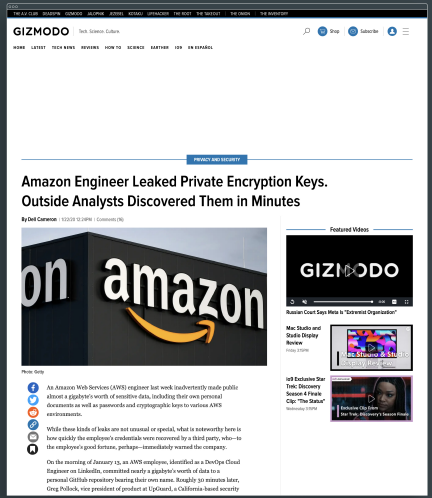
Previous: Tracking phones without a warrant ruled unc... Next: Monday review – Adobe botches, Apache bleeds a...

by Paul Ducklin

Finnish security researcher Juho Nurminen is a bit of a retweet celebrity right now, for all the wrong reasons.

Not his wrong reasons, but the wrong reasons of Adobe's Product Security Incident Response Team (PSIRT).

To explain.



GIZMODO Tech. Science. Culture.

Shop Subscribe

HOME LATEST TECH KINGS REVIEWS HOW TO SCIENCE EARTHER ICYMI EN ESPAÑOL

PRIVACY AND SECURITY

Amazon Engineer Leaked Private Encryption Keys. Outside Analysts Discovered Them in Minutes

By Dell Cameron | 9/23/18 12:29PM | Comments (16)

Photo: Getty

An Amazon Web Services (AWS) engineer last week inadvertently made public almost a gigabyte's worth of sensitive data, including their own personal documents as well as passwords and cryptographic keys to various AWS environments.

While these kinds of leaks are not unusual or special, what is noteworthy here is how quickly the employee's credentials were recovered by a third party, who—to the employee's good fortune, perhaps—immediately warned the company.

On the morning of January 13, an AWS employee, identified as a DevOps Cloud Engineer on LinkedIn, committed nearly a gigabyte's worth of data to a personal GitHub repository bearing their own name. Roughly 30 minutes later, Greg Pollock, vice president of product at UpGuard, a California-based security

Featured Videos

Russian Court Says Meta Is "Extremist Organization"

Mac Studio and Studio Display Review

Friday 2:00PM

Is Sharing a Shelf Worth It?

Is9 Exclusive Star Trek: Discovery Season 4 Finale Clip: "The Statue"

Wednesday 3:00PM

Exclusive Clip Intro

Star Trek: Discovery's Season Finale

https://nakedsecurity.sophos.com/2017/09/23/adobe-security-team-posts-public-key-together-with-private-key; https://gizmodo.com/amazon-engineer-leaked-private-encryption-keys-outside-1841160934

Forward Security to Mitigate Key Leakage

- Key leakage: severe issue for encryption schemes – all data immediately in danger

Forward Security to Mitigate Key Leakage

- Key leakage: severe issue for encryption schemes – all data immediately in danger
- Mitigation: forward security guarantees that old data is still safe

Forward Security to Mitigate Key Leakage

- Key leakage: severe issue for encryption schemes – all data immediately in danger
- Mitigation: forward security guarantees that old data is still safe
- Mandatory in TLS 1.3, recognized by industry: Apple, Cloudflare, Google, Microsoft, ...

Forward Security to Mitigate Key Leakage

- Key leakage: severe issue for encryption schemes – all data immediately in danger
- Mitigation: forward security guarantees that old data is still safe
- Mandatory in TLS 1.3, recognized by industry: Apple, Cloudflare, Google, Microsoft, ...
- Broadly considered in several cryptographic applications: e.g., key exchange [Gün90, DDG+20, BRS23, RSS23], (updatable) public-key encryption [CHK03, Gro21, DKW21], signatures [BM91, DGNW20], searchable encryption [BMO17], Cloud backup [DCM20], proxy cryptography [DKLRSS17], Tor [LGM+17], content delivery networks [DRSS21]

Forward Security to Mitigate Key Leakage

- Key leakage: severe issue for encryption schemes – all data immediately in danger
- Mitigation: forward security guarantees that old data is still safe
- Mandatory in TLS 1.3, recognized by industry: Apple, Cloudflare, Google, Microsoft, ...
- Broadly considered in several cryptographic applications: e.g., key exchange [Gün90, DDG+20, BRS23, RSS23], (updatable) public-key encryption [CHK03, Gro21, DKW21], signatures [BM91, DGNW20], searchable encryption [BMO17], Cloud backup [DCM20], proxy cryptography [DKLRSS17], Tor [LGM+17], content delivery networks [DRSS21]

Goal in this work: enhancing forward-security paradigm to updatable encryption. Not known to be achievable at all before.

Updatable Encryption

- Updatable encryption is a symmetric primitive that allows to rotate encryption keys in (outsourced) ciphertexts without the need to download, decrypt, re-encrypt and upload already encrypted data

Updatable Encryption

- Updatable encryption is a symmetric primitive that allows to rotate encryption keys in (outsourced) ciphertexts without the need to download, decrypt, re-encrypt and upload already encrypted data
- Key generation: $K_1 \leftarrow \mathsf{Gen}$

Updatable Encryption

- Updatable encryption is a symmetric primitive that allows to rotate encryption keys in (outsourced) ciphertexts without the need to download, decrypt, re-encrypt and upload already encrypted data
- Key generation: $K_1 \leftarrow \mathsf{Gen}$
- Encryption: $C_1 \leftarrow \mathsf{Enc}(K_1, M)$

Updatable Encryption

- Updatable encryption is a symmetric primitive that allows to rotate encryption keys in (outsourced) ciphertexts without the need to download, decrypt, re-encrypt and upload already encrypted data
- Key generation: $K_1 \leftarrow \mathsf{Gen}$
- Encryption: $C_1 \leftarrow \mathsf{Enc}(K_1, M)$
- Next key: $(K_2, \Delta_2) \leftarrow \mathsf{NextKey}(K_1)$

Updatable Encryption

- Updatable encryption is a symmetric primitive that allows to rotate encryption keys in (outsourced) ciphertexts without the need to download, decrypt, re-encrypt and upload already encrypted data
- Key generation: $K_1 \leftarrow \mathsf{Gen}$
- Encryption: $C_1 \leftarrow \mathsf{Enc}(K_1, M)$
- Next key: $(K_2, \Delta_2) \leftarrow \mathsf{NextKey}(K_1)$
- Update: $C_2 \leftarrow \mathsf{Update}(C_1, \Delta_2)$ with property $M = \mathsf{Dec}(K_2, C_2)$

Updatable Encryption

- Updatable encryption is a symmetric primitive that allows to rotate encryption keys in (outsourced) ciphertexts without the need to download, decrypt, re-encrypt and upload already encrypted data
- Key generation: $K_1 \leftarrow \mathsf{Gen}$
- Encryption: $C_1 \leftarrow \mathsf{Enc}(K_1, M)$
- Next key: $(K_2, \Delta_2) \leftarrow \mathsf{NextKey}(K_1)$
- Update: $C_2 \leftarrow \mathsf{Update}(C_1, \Delta_2)$ with property $M = \mathsf{Dec}(K_2, C_2)$
- Security: distinguish ciphertexts for a chosen epoch, given access to keys and tokens

Updatable Encryption

- Updatable encryption is a symmetric primitive that allows to rotate encryption keys in (outsourced) ciphertexts without the need to download, decrypt, re-encrypt and upload already encrypted data
- Key generation: $K_1 \leftarrow \mathsf{Gen}$
- Encryption: $C_1 \leftarrow \mathsf{Enc}(K_1, M)$
- Next key: $(K_2, \Delta_2) \leftarrow \mathsf{NextKey}(K_1)$
- Update: $C_2 \leftarrow \mathsf{Update}(C_1, \Delta_2)$ with property $M = \mathsf{Dec}(K_2, C_2)$
- Security: distinguish ciphertexts for a chosen epoch, given access to keys and tokens

Observation: forward security cannot be achieved when all tokens leak

Starting Building Block: Symmetric Puncturable Encryption (SPE)

- Tag-based encryption scheme with fine-grained forward security (PE as a general paradigm considered already in, e.g., [GM15,GHJL18,DGJSS21,AGJ21,...])

Starting Building Block: Symmetric Puncturable Encryption (SPE)

- Tag-based encryption scheme with fine-grained forward security (PE as a general paradigm considered already in, e.g., [GM15,GHJL18,DGJSS21,AGJ21,...])
- Key generation: $K_\varepsilon \leftarrow \mathsf{KeyGen}$

Starting Building Block: Symmetric Puncturable Encryption (SPE)

- Tag-based encryption scheme with fine-grained forward security (PE as a general paradigm considered already in, e.g., [GM15,GHJL18,DGJSS21,AGJ21,...])
- Key generation: $K_\varepsilon \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{\{t_1\}} \leftarrow \mathsf{Enc}(K_\varepsilon, t_1, M)$

Starting Building Block: Symmetric Puncturable Encryption (SPE)

- Tag-based encryption scheme with fine-grained forward security (PE as a general paradigm considered already in, e.g., [GM15,GHJL18,DGJSS21,AGJ21,...])
- Key generation: $K_\varepsilon \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{\{t_1\}} \leftarrow \mathsf{Enc}(K_\varepsilon, t_1, M)$
- Puncturing: $K_{\{t_1\}} \leftarrow \mathsf{Punc}(K_\varepsilon, t_1)$

Starting Building Block: Symmetric Puncturable Encryption (SPE)

- Tag-based encryption scheme with fine-grained forward security (PE as a general paradigm considered already in, e.g., [GM15,GHJL18,DGJSS21,AGJ21,...])
- Key generation: $K_\varepsilon \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{\{t_1\}} \leftarrow \mathsf{Enc}(K_\varepsilon, t_1, M)$
- Puncturing: $K_{\{t_1\}} \leftarrow \mathsf{Punc}(K_\varepsilon, t_1)$
- Properties: $K_{\{t_1\}}$ no longer useful to decrypt ciphertexts associated to t_1 (such as $C_{\{t_1\}}$), but still all others with $t_2, \ldots$

Starting Building Block: Symmetric Puncturable Encryption (SPE)

- Tag-based encryption scheme with fine-grained forward security (PE as a general paradigm considered already in, e.g., [GM15,GHJL18,DGJSS21,AGJ21,...])
- Key generation: $K_\varepsilon \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{\{t_1\}} \leftarrow \mathsf{Enc}(K_\varepsilon, t_1, M)$
- Puncturing: $K_{\{t_1\}} \leftarrow \mathsf{Punc}(K_\varepsilon, t_1)$
- Properties: $K_{\{t_1\}}$ no longer useful to decrypt ciphertexts associated to t_1 (such as $C_{\{t_1\}}$), but still all others with $t_2, \ldots$
- Distinguishing feature: repeated puncturing of secret keys (add more tags to the key, exclude more ciphertexts from being decryptable)

Starting Building Block: Symmetric Puncturable Encryption (SPE)

- Tag-based encryption scheme with fine-grained forward security (PE as a general paradigm considered already in, e.g., [GM15,GHJL18,DGJSS21,AGJ21,...])
- Key generation: $K_\varepsilon \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{\{t_1\}} \leftarrow \mathsf{Enc}(K_\varepsilon, t_1, M)$
- Puncturing: $K_{\{t_1\}} \leftarrow \mathsf{Punc}(K_\varepsilon, t_1)$
- Properties: $K_{\{t_1\}}$ no longer useful to decrypt ciphertexts associated to t_1 (such as $C_{\{t_1\}}$), but still all others with $t_2, \ldots$
- Distinguishing feature: repeated puncturing of secret keys (add more tags to the key, exclude more ciphertexts from being decryptable)
- Security: distinguish ciphertexts on a chosen tag t, even if punctured key on t is available

Special Case of SPE: Epoch-Based Puncturable Encryption

- Stepping back: SPE with linearly evolving epochs $e_1, e_2, \ldots$ encoded in keys

Special Case of SPE: Epoch-Based Puncturable Encryption

- Stepping back: SPE with linearly evolving epochs $e_1, e_2, \ldots$ encoded in keys
- Key generation: $\underline{K_{e_1}} \leftarrow \mathsf{KeyGen}$

Special Case of SPE: Epoch-Based Puncturable Encryption

- Stepping back: SPE with linearly evolving epochs $e_1, e_2, \ldots$ encoded in keys
- Key generation: $\underline{K_{e_1}} \leftarrow \mathsf{KeyGen}$
- Encryption: returns $\underline{C_{e_1}} \leftarrow \mathsf{Enc}(\underline{K_{e_1}}, M)$

Special Case of SPE: Epoch-Based Puncturable Encryption

- Stepping back: SPE with linearly evolving epochs $e_1, e_2, \ldots$ encoded in keys
- Key generation: $\underline{K_{e_1}} \leftarrow \mathsf{KeyGen}$
- Encryption: returns $\underline{C_{e_1}} \leftarrow \mathsf{Enc}(\underline{K_{e_1}}, M)$
- Puncturing: returns $\underline{K_{e_2}} \leftarrow \mathsf{Punc}(\underline{K_{e_1}})$

Special Case of SPE: Epoch-Based Puncturable Encryption

- Stepping back: SPE with linearly evolving epochs $e_1, e_2, \ldots$ encoded in keys
- Key generation: $\underline{K_{e_1}} \leftarrow \mathsf{KeyGen}$
- Encryption: returns $\underline{C_{e_1}} \leftarrow \mathsf{Enc}(\underline{K_{e_1}}, M)$
- Puncturing: returns $\underline{K_{e_2}} \leftarrow \mathsf{Punc}(\underline{K_{e_1}})$
- Properties: K_{e_2} no longer useful to decrypt ciphertexts associated to $\underline{K_{e_1}}$

Special Case of SPE: Epoch-Based Puncturable Encryption

- Stepping back: SPE with linearly evolving epochs $e_1, e_2, \ldots$ encoded in keys
- Key generation: $\underline{K_{e_1}} \leftarrow \mathsf{KeyGen}$
- Encryption: returns $\underline{C_{e_1}} \leftarrow \mathsf{Enc}(\underline{K_{e_1}}, M)$
- Puncturing: returns $\underline{K_{e_2}} \leftarrow \mathsf{Punc}(\underline{K_{e_1}})$
- Properties: K_{e_2} no longer useful to decrypt ciphertexts associated to $\underline{K_{e_1}}$

Observation: coarse-grained puncturing on epochs, can we make it more fine-grained?

Contribution: Tag-Inverse Puncturable Encryption (TIPE)

- Idea: "exclude" certain ciphertexts from epoch-based puncturing (i.e., partly inverts the concept of tags in PE) and add expiry epochs

Contribution: Tag-Inverse Puncturable Encryption (TIPE)

- Idea: "exclude" certain ciphertexts from epoch-based puncturing (i.e., partly inverts the concept of tags in PE) and add expiry epochs
- Key generation: $K_{e_1} \leftarrow \mathsf{KeyGen}$

Contribution: Tag-Inverse Puncturable Encryption (TIPE)

- Idea: "exclude" certain ciphertexts from epoch-based puncturing (i.e., partly inverts the concept of tags in PE) and add expiry epochs
- Key generation: $K_{e_1} \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{e_1, \underline{t_1, exp}} \leftarrow \mathsf{Enc}(K_{e_1}, M, \underline{t_1, exp})$

Contribution: Tag-Inverse Puncturable Encryption (TIPE)

- Idea: "exclude" certain ciphertexts from epoch-based puncturing (i.e., partly inverts the concept of tags in PE) and add expiry epochs
- Key generation: $K_{e_1} \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{e_1, \underline{t_1, exp}} \leftarrow \mathsf{Enc}(K_{e_1}, M, \underline{t_1, exp})$
- Key puncturing: $(K_{e_2}, \underline{\Delta_{e_2, t_1}, \ldots}) \leftarrow \mathsf{KPunc}(K_{e_1}, \underline{t_1, \ldots})$

Contribution: Tag-Inverse Puncturable Encryption (TIPE)

- Idea: "exclude" certain ciphertexts from epoch-based puncturing (i.e., partly inverts the concept of tags in PE) and add expiry epochs
- Key generation: $K_{e_1} \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{e_1,t_1,exp} \leftarrow \mathsf{Enc}(K_{e_1}, M, t_1, exp)$
- Key puncturing: $(K_{e_2}, \Delta_{e_2,t_1}, \ldots) \leftarrow \mathsf{KPunc}(K_{e_1}, t_1, \ldots)$
- Exclude from puncturing: $C_{e_2,t_1,exp} \leftarrow \mathsf{ExPunc}(C_{e_1,t_1,exp}, \Delta_{e_2,t_1})$

Contribution: Tag-Inverse Puncturable Encryption (TIPE)

- Idea: "exclude" certain ciphertexts from epoch-based puncturing (i.e., partly inverts the concept of tags in PE) and add expiry epochs
- Key generation: $K_{e_1} \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{e_1,\underline{t_1,exp}} \leftarrow \mathsf{Enc}(K_{e_1}, M, \underline{t_1, exp})$
- Key puncturing: $(K_{e_2}, \underline{\Delta_{e_2,t_1}, \ldots}) \leftarrow \mathsf{KPunc}(K_{e_1}, \underline{t_1, \ldots})$
- Exclude from puncturing: $\underline{C_{e_2,t_1,exp} \leftarrow \mathsf{ExPunc}(C_{e_1,t_1,exp}, \Delta_{e_2,t_1})}$
- Properties: $M = \mathsf{Dec}(K_{e_2}, \underline{C_{e_2,t_1,exp}})$ if $\underline{exp \geq 2}$

Contribution: Tag-Inverse Puncturable Encryption (TIPE)

- Idea: "exclude" certain ciphertexts from epoch-based puncturing (i.e., partly inverts the concept of tags in PE) and add expiry epochs
- Key generation: $K_{e_1} \leftarrow \mathsf{KeyGen}$
- Encryption: $C_{e_1,t_1,exp} \leftarrow \mathsf{Enc}(K_{e_1}, M, t_1, exp)$
- Key puncturing: $(K_{e_2}, \Delta_{e_2,t_1}, \ldots) \leftarrow \mathsf{KPunc}(K_{e_1}, t_1, \ldots)$
- Exclude from puncturing: $C_{e_2,t_1,exp} \leftarrow \mathsf{ExPunc}(C_{e_1,t_1,exp}, \Delta_{e_2,t_1})$
- Properties: $M = \mathsf{Dec}(K_{e_2}, C_{e_2,t_1,exp})$ if $exp \geq 2$
- Security: distinguish ciphertexts for a chosen epoch, even if *all* keys and tokens are available (except for trivial wins)

Application: Updatable Encryption with Expiring Ciphertexts

- TIPE is a special case of updatable encryption with expiring ciphertext (latter also defined in our work)

Application: Updatable Encryption with Expiring Ciphertexts

- TIPE is a special case of updatable encryption with expiring ciphertext (latter also defined in our work)
- Security: our defined UE notion implies CPA notions of common UE schemes

Application: Updatable Encryption with Expiring Ciphertexts

- TIPE is a special case of updatable encryption with expiring ciphertext (latter also defined in our work)
- Security: our defined UE notion implies CPA notions of common UE schemes
- Novel: first backwards-leak UE scheme with sub-linear ciphertexts from standard assumptions (solves open problem posed in [GP23, MPW23])

Application: Updatable Encryption with Expiring Ciphertexts

- TIPE is a special case of updatable encryption with expiring ciphertext (latter also defined in our work)
- Security: our defined UE notion implies CPA notions of common UE schemes
- Novel: first backwards-leak UE scheme with sub-linear ciphertexts from standard assumptions (solves open problem posed in [GP23, MPW23])
- Glimpse of techniques: novel adaptation of dual system groups [CW13, GCTC16]

Takeaways

- Forward security is an essential security feature of many cryptographic primitives

Takeaways

- Forward security is an essential security feature of many cryptographic primitives
- Tag-Inverse Puncturable Encryption (TIPE) offers a novel view on new applications areas such as updatable encryption

Takeaways

- Forward security is an essential security feature of many cryptographic primitives
- Tag-Inverse Puncturable Encryption (TIPE) offers a novel view on new applications areas such as updatable encryption
- Open: use TIPE for applications beyond UE?

AUSTRIAN INSTITUTE OF TECHNOLOGY

Thank you!

FWF
Der Wissenschaftsfonds.