Analisis *Attack Surface* Infrastruktur Kunci Publik

Muhammad Gema Akbar 13510099[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]mgemaakbar@students.itb.ac.id

***Abstract*—Infrastruktur kunci public adalah salah satu penerapan kriptografi. Infrastruktur kunci public digunakan salah satunya dalam aplikasi web. Beberapa insiden penyerangan membuktikan bahwa infrastruktur kunci public dapat diserang. Serangan yang berhasil dapat mengakibatkan kebocoran informasi yang seharusnya rahasia dan bahkan modifikasi dari informasi yang terkirim. Makalah ini akan membahas *attack surface* atau bagian/titik dari infrastruktur kunci public mana sajakah yang dapat menjadi objek serangan.**

***Index Terms*—analisis, serangan, *interception proxy*, SSL,TLS, kerentanan.**

I. Infrastruktur Kunci Publik

Dalam kriptografi, infrastruktur kunci public adalah teknik yang dapat membuat seseorang untuk berkomunikasi secara aman pada suatu jaringan public yang tidak aman, dan dengan menggunakan infrastruktur kunci public juga kita dapat mengetahui keaslian identitas dari seseorang dengan menggunakan tandatangan digital.

Infrastruktur kunci public dapat membuat, menyimpan, dan mendistribusikan sertifikat digital yang digunakan untuk memverifikasi bahwa suatu kunci public dimiliki oleh suatu pihak. Infrastruktur kunci public membuat sertifikat digital yang memetakan kunci public pada entitas, menyimpan sertifikat dalam suatu penyimpanan dan melakukan revokasi.

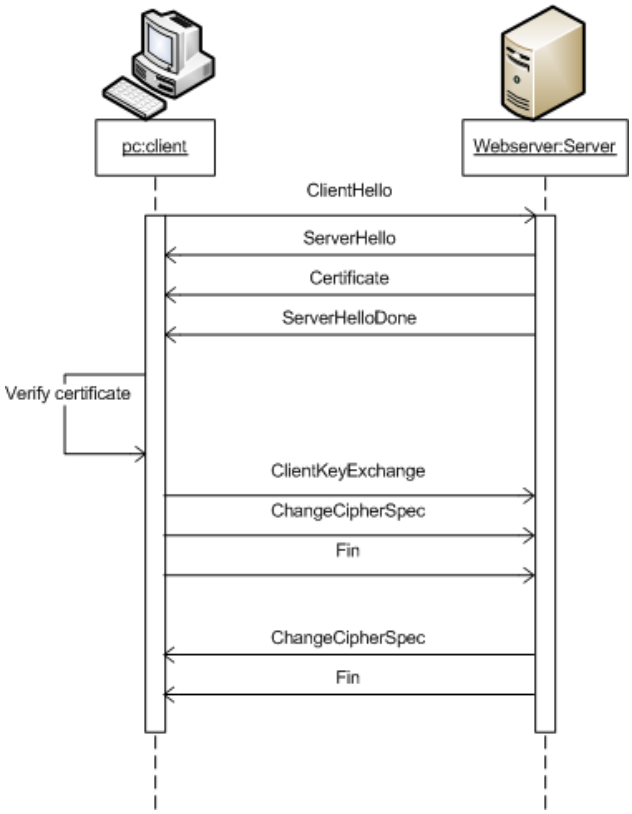
Infrastruktur kunci public terdiri dari:

- *Certificate Authority* (CA) yang mengeluarkan dan memverifikasi sertifikat digital
- *Registration authority* yang memverifikasi identitas pengguna yang meminta informasi dari CA
- *Central directory* , suatu lokasi untuk menyimpan kunci
- *Certificate management system*
- *Certificate policy*

SSL (*Secure Socket Layer*) dan TLS (*Transport Layer Security*) memiliki mekanisme yang sama. Netscape awalnya mengajukan SSL sebagai spesifikasi draft v2.0 di tahun 1994. SSL v2.0 diperbaharui menjadi SSL v3.0 yang merupakan dasar untuk TLS v1.0. TLS v1.0 telah diperbaharui menjadi TLS v.1.1 dan TLS v.1.2. SSL dan TLS adalah dua istilah yang sering digunakan secara bergantian untuk mengacu pada spesifikasi TLS v1.1.[1]

Berikut adalah SSL *handshake:*

1



Pic 1. SSL *Handshake*

SSL *handshake* memiliki guna sebagai berikut:

- Pertukaran informasi kompatibilitas versi dan *extension*
- Pertukaran kunci dan sertifikat yang berisi identitas
- Penentuan *cipher suite* yang digunakan untuk enkripsi
- Membangun *transport layer* yang aman untuk digunakan oleh *layer* yang lebih tinggi

TLS tidak melakukan validasi terhadap identitas dari *endpoint.* Ini adalah tugas X.509.

Server dapat meminta klien untuk menyediakan sertifikat identitasnya sendiri melalui pesan sertifikat klien.

Bagian dari SSL yang menunjukan keaslian ada pada kemampuan untuk memvalidasi sertifikat dari suatu *endpoint.* Spesifikasi X.509 mengatur format dari sertifikat tersebut, dan juga pertimbangan-pertimbangan yang penting untuk memvalidasi keutuhannya.[2]

RFC5280 berisi detail dari spesifikasi tersebut, tapi proses validasi intinya terdiri dari poin-poin berikut:

- Validasi tandatangan digital dari suatu sertifikat
- Ikuti rantai sertifikat untuk menentukan CA yang relevan
- Menentukan jika *root* CA dapat dipercaya
- Mengkonfirmasi vailiditas sertifikat *temporal* dengan cara mengecek tanggal sertifikat dibuat dan kapan sertifikat tidak berlaku lagi
- Menentukan untuk subjek apa sertifikat berlaku
- Menentukan apakah sertifikat telah dilakukan revokasi

Dalan spesifikasi X.509, cara melakukan revokasi sertifikat adalah melalui *certificate revocation list (*CRL*).* Tapi spesifikasi tersebut meninggalkan ruang bagi *extension* lain untuk memperluas cara. Salah satu *extension* yang ada adalah Online Certificate Status Protocol (OCSP).

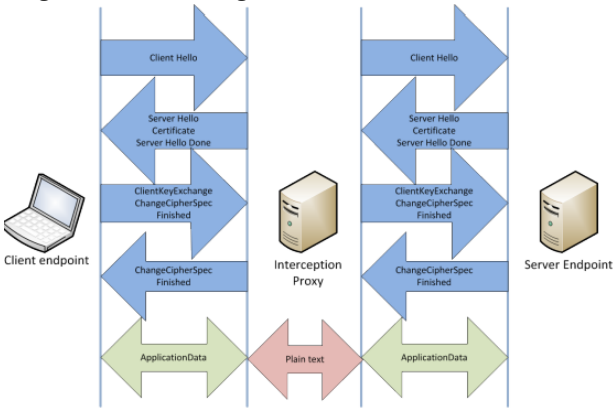
Kegagalan dalam validasi sertifikat dapat menyebabkan rusaknya keaslian dari SSL. Rusaknya sifat keaslian dari SSL dapat menyebabkan pemalsuan identitas, dan serangan terhadap keutuhan/integritas, antar *client-server*.

II. SERANGAN TERHADAP *INTERCEPTION PROXY*

Untuk memeriksa isi plainteks dari komunikasi SSL, yang disebut dengan *interception proxy* menyelipkan dirinya dalam alur lalulintas dan menhentikan rekues klien. *Interception proxy* adalah perantara antar klien dan server. *Interception proxy* membuat rekues kedua atas nama klien ke server, kejadian ini menyebabkan *session* dari klien ke server menjadi dua *session* yang terpisah, yaitu klien ke *proxy* dan *proxy* ke server, walaupun saling terkait.

Interception proxy dapat diimplementasikan dengan banyak cara, tergantung dari tujuan dan tipe pemeriksaan terhadap lalulintas yang dilakukan. Walaupun memliki jenis yang berbeda-beda, *interception proxy* dapat dianggap sebagai titik yang melakukan MITM.

Agar dapat bertidak sebagai server untuk SSL *session* dari client, *interception proxy* harus memliki akses ke kunci privat yang terkait dengan sertifikat yang ditangani. Karena kunci privat *endpoint* tidak ada, *interception rpoxy* harus dapat meng*generate* sertifikat baru dan pasangan kunci untuk digunakan *session.*



Pic 2. Posi*si interception proxy* dalam koneksi *client-server*

Sertifikat harus ditandatangani oleh CA yang dipercaya oleh klien. Jika tidak, gagal validasi dapat terjadi pada klien. Ada dua syarat klien dapat percaya dengan sertifikat:[3]

- *Private CA*
- *Public SubCA*

Dalam *private CA.* sertifikat yang identic dengan suatu CA disebarkan ke titik ujung (*endpoint*) , yang nantinya menggunakan sertifikat sebagai *root* yang dipercaya.

Dalam *public SubCA,* CA memberikan izin pada suatu pihak untuk melakukan tandatangan digital.

Penggunaan *interception proxy* dapat meningkatkan resiko serangan, karena *interception proxy* memiliki akses plainteks dari suatu data yang telah terenkripsi. Maka dari itu penyerang akan memiliki pikiran bahwa menyerang *interception proxy* adalah target yang tepat jika penyerang ingin mencuri plainteks. Ditambhah lagi, jika kita bisa mencuri kunci tandatangan CA dari *interception proxy,* kita bisa memalsukan komunikasi Antara client-server.

Resiko lain adalah berkurangnya kekuatan cipher yang digunakan. Penggunaan *interception proxy* dapat menghasilkan sesi terenkripsi Antara titik client-server. Karena sesi menegosiasikan cipher suite secara independen, ada kemungkinan bahwa salah satu dari server atau bahkan keduanya menggunakan cipher suite lemah.

Sekarang akan dibahas beberapa serangan yang mungkin untuk dilakukan terhadap *interception proxy. Interception proxy* dapat diserang dengan menggunakan celah *BasicContraint. BasicConstraint* adalah salah satu atribut sertifikat, jika *basicconstraint* bernilai *false,* maka sertifikat tersebut bukan berasal dari CA perantara yang valid. Beberapa browser dapat jatuh pada kerentanan ini.

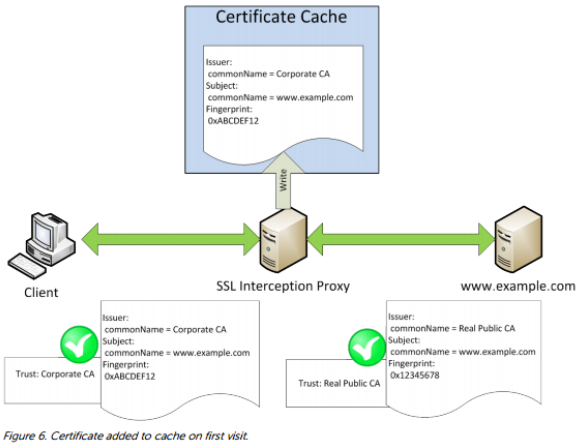
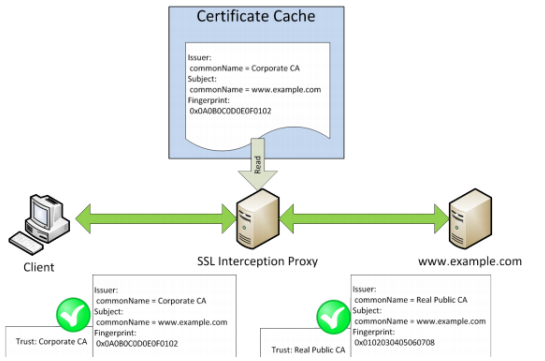


Figure 6. Certificate added to cache on first visit.

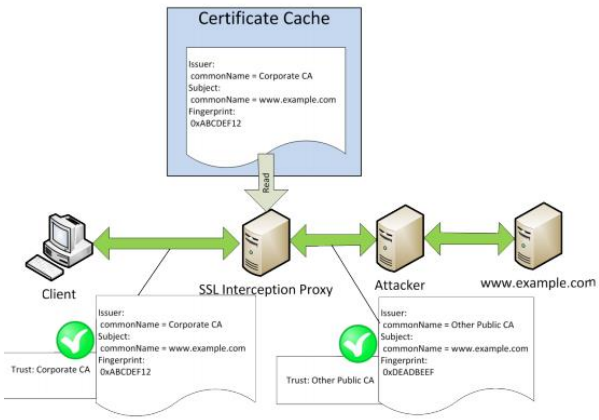
Pic 3 Caching yang benar

Serangan lainnya adalah dengan memanfaatkan kesalahan dalam *caching* dari verifikasi sertifikat. Dalam melakukan tugasnya, *interception proxy* mungkin tidak membuat session *client-server* baru demi meningkatkan kinerja. *Caching* sertifikat dapat memberikan kesempatan bagi penyerang.



Pic 4 Pengambilan sertifikat dari cache

Tergantung cara yang digunakan untuk mengindeks dan mengambil sertifikat dari cache, seorang penyerang dapat memasukan sertifikat pada sesi server-side. Jika ini terjadi klien tidak akan dapat menyadari bahwa komunikasi dengan server dapat dilakuan semacam MITM.



Pic 5. MITM dengan sertifikat palsu

III. SERANGAN TERHADAP SSL/TLS

SSL yang bertanggung jawab dalam enkripsi lalulintas komunikasi adalah satu objek serangan yang jelas, Di bagian makalah berikut ini akan dijelaskan beberapa serangan-serangan yang mungkin dilakukan terhadap SSL/TLS.

BEAST adalah salah satu serangan terhadap SSL/TLS. Ketika kita melakukan login ke halaman HTTPS, kita dapat meilhat session-ID. *Session* ID adalah angka acak atau string acak yang diberikan oleh server website ke browser yang digunakan client. *Session ID* dapat ditemaukan dalam *cookie* atau di URL web browser. Biasanya session ID dienkripsi untuk mencegah pe mbajakan dari *session.*

BEAST memiliki cara kerja sebagai berikut. Pertama, penyerang mengirim Javascript untuk dijalankan pada mesin korban, dapat dilakukan dengan CSRF, rekayasa social, dan lain lain. Script tersebut berjalan pada computer korban dan dapat menangkap semua informasi pada *header* dan *cookie* yang telah dienkripsi. Kedua, SSL/TLS dapat melakukan enkripsi data dengan dua jenis cipher, block cipher: AES dan DES, atau Stream cipher: RC4 misalnya. TLS 1.0 menggunakan block cipher. Jika kita memiliki dua plainteks yang identik, maka kita memiliki cipherteks yang memilki pola yang sama.

CRIME (Compression Ratio Info-leak Made Easy) adalah salah satu serangan terhadap SSL/TLS, memiliki cara kerja dengan melakukan *side-channel* terhadap *compression* yang digunakan lalulintas HTTPS.

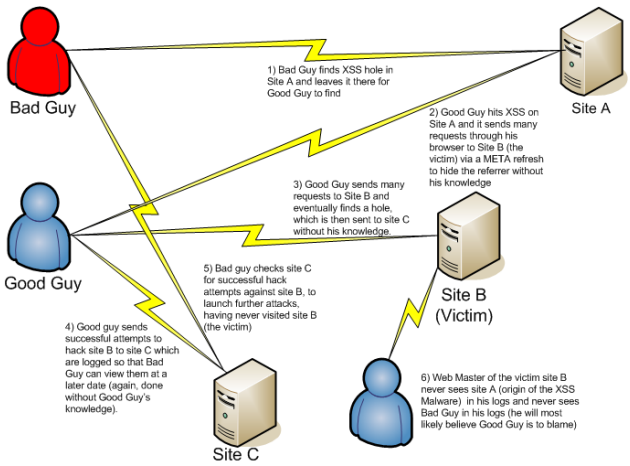
Penyerang yang melakukan CRIME dapat melakukan:

- Memasukan plainteks ke HTTP request korban
- Mengukur ukuran dari lalulintas komunikasi yang terenkripsi

Seseorang dapat menjadi korban serangan CRIME jika:

- Korban ada dalam satu WLAN yang sama dengan penyerang
- Penyerang telah memiliki akses pada router yang digunakan korban
- Penyerang adalah admin jaringan, ISP yang digunakan oleh korban.
- Korban mengunjungi situs yang telah dibuat sedemikian rupa oleh penyerang

C pada singkatan CRIME berarti *compression.* Kompresi membuat kita dapat mengirimkan banyaknya data yang sama tapi dengan banyak bit yang lebih sedikit. DEFLATE adalah salah satu kompresi di internet.



Pic 6 Diagram CRIME

CRIME memliki cara kerja dengan cara mencuri *cookie* dari suatu HTTPS *session* yang telah dibangun korban, *cookie* dicuri dengan cara mengakali browser agar mengirim rekues yang telah terkompresi dan memanfaatkan informasi yang bocor ketika pengiriman. Perbedaan ukuran dari pesan yang terkompresi dapat membantu untuk menentukan isi dari *cookie*.

Sistem yang rentan terhadap CRIME:

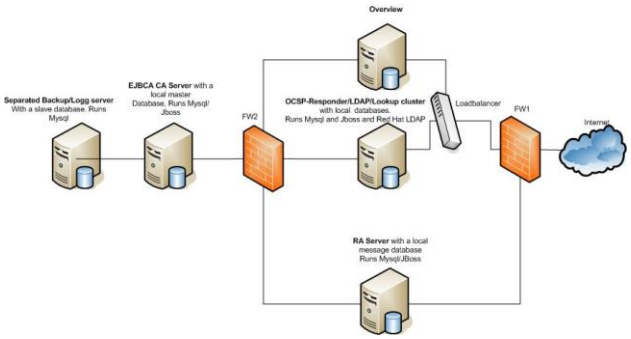
- TLS 1.0
- SPDY
- Aplikasi yang menggunakan *compression* pada TLSnya

Serangan *null prefix* terhadap sertifikat SSL/TLS adalah salah satu serangan terhadap sertifikat SSL/TLS yang disebabkan oleh kesalahan dalam implementasi SSL/TLS dalam suatu perangkat lunak yang menggunakan SSL/TLS. Kesalahan ini memungkinkan seseorang untuk membuat sertifikat palsu dan melakukan *man-in-the-middle* terhadap traffic yang telah terenkripsi dengan sertifikat palsu yang berkaitan. Cara kerja serangan ini adalah, jika kita melakukan *Certificate Signing Request* seperti ini: www.google.com\0.satuwebsite.com CA akan mengabaikan apa yang ada sebelum \0 dan hanya akan melihat *root domain* yaitu satuwebsite.com. Jika kita adalah pemilik satuwebsite.com kita akan dapat membuktikan kepemilikan domain pada CA. Hal ini dapat menjadi bahan serangan karena banyak dari implementasi SSL/TLS menganggap *field* yang diambil dari sertifikat x509 sebagai *string* C biasa. Akibatnya perbandingan Antara www.google.com\0satuwebsite.com akan dianggap sama dengan www.google.com. Dan kita dapat berbohong pada mekanisme validasi karena mekanisme validasi akan menganggap satuwebsite.com adalah domain yang valid.

III. SERANGAN TERHADAP OCSP

OCSP (*Online Certificate Status Protocol*) dikembangkan untuk mendukung *certificate revocation* di dunia dimana lebih banyak sertifikat dikeluarkan oleh CA. Ketika lebih banyak revokasi harus dilakukan, metode lama dimana dilakukan pencatatan *certificate revocation list* yang harus diperbaharui secara manual untuk setiap CA tidak dapat dilakukan secara praktis lagi.

OCSP dibuat untuk mendukung rekues dari klien yang diberi sertifikat untuk mengecek validitas dari sertifikat-sertifikat tersebut secara *real-time*. *Browser* seperti Firefox ketika diberi sertifikat www.google.com yang belum pernah dilihat sebelumnya, sebelum Firefox menerima sertifikatnya, Firefox melakukan koneksi ke pihak yang mengeluarkan sertifikat dan bertanya apakah certifikat masih dapat dianggap valid.



Pic 7. Sistem OCSP

Respon dari rekues OCSP memilki struktur seperti ini:

```
  OCSPResponse ::= SEQUENCE {
  responseStatus OCSPResponseStatus,
  responseBytes       [0Ł       EXPLICIT
ResponseBytes OPTIONAL
}
```

Ini adalah struktur ReponseBytes:

```
  BasicOCSPResponse ::= SEQUENCE {
  tbsResponseData ResponseData,
  signatureAlgorithm AlgorithmIdentier,
signature BIT STRING,

erts [0Ł EXPLICIT SEQUENCE OF Certi□
ate OPTIONAL
}
```

Kita lihat bahwa dalam struktur ResponseBytes ada *signature* oleh CA yang akan diminta *query*.

Coba kita lihat struktur ResponseStatus, terlihat bahwa ada beberapa kemungkinan untuk pilihan status:

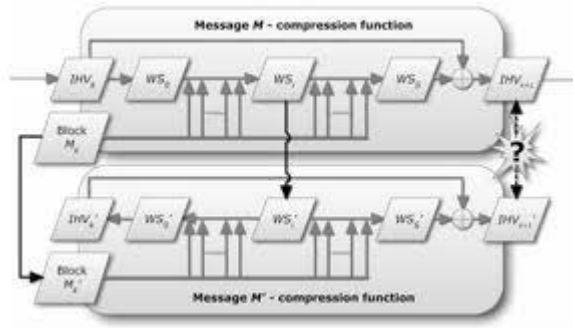
```
  OCSPResponseStatus ::= ENUMERATED {
  su

essful (0), □Response has valid
on□rmations
  malformedRequest (1), □Illegal
on□rmation request
  internalError (2), □Internal error in
```

```
issuer
  tryLater (3), Try again later
  sigRequired (5), Must sign the
request
  unauthorized (6) Request
unauthorized
}
```

Jika kita ingin memalsukan respon OCSP, yang mungkin dilakukan adalah dengan memalsukan respon dengan status TryLater. Ini disebabkan oleh status TryLater tidak mengharuskan kita untuk membuat semacam *signature*. Serangan ini dapat berhasil karena banyak dari implementasi OCSP akan menerima respon tanpa memberikan peringatan kepada user bahwa ada yang salah.

III. Serangan Terhadap X.509

Serangan terhadap X509 salah satunya dapat dilakukan dengan memanfaatkan kelemahan dalam fungsi hash yang memungkinkan pembuatan pesan yang berbeda tapi memiliki nilai Md5 hash yang sama. Fenomena ini sering disebut dengan *hash collision.* Fenomena ini dapat dimanfaatkan untuk melakukan beberapa scenario serangan yang realistis dan mungkin untuk dilakukan di dunia nyata (tidak hanya teori).



Pic 4. *Chosen-prefix hash collision*

Serangan ini dapat memiliki hasil dengan dibuatnya sertifikat palsu dari suatu CA dan sertifikat tersebut akan diterima sebagai sertifikat valid dan dipercaya oleh browser-browser, karena sertifikat dibuat seolah-olah ditandatangani oleh salah satu *root CA* yang dipercaya oleh browser. Maka sertifikat website apapun yang ditandatangani oleh CA palsu kita akan dipercaya juga. Jika seorang pengguna adalah korban dari serangan MITM yang menggunakan sertifikat palsu tersebut, korban tidak akan dapat mengetahui bahwa ia sedang menjadi korban MITM karena status sertifikat berbohong kepada korban bahwa sertifikatnya baik-baik saja.

Jika serangan berhasil dilakukan, penyerang dapat mengetahui lalulintas dan bahkan melakukan modifikasi terhadap lalulintas data. Infrastruktur CA dibuat untuk mencegah serangan seperti ini.

Skenario serangan dapat dilakukan sebagai berikut. Karena algoritma tandatangan digital tidakdapat menandatangani data dengan jumlah besar secara efisien, banyak dari implementasi menggunakan fungsi hash untuk mengurangi besarnya data yang harus ditandatangani, agar ukurannya menjadi konstan. Skema tandatangan digital sering rentan terhadap hash collision kecuali digunakan teknik randomized hashing.

Sertifikat kunci public sepertihalnya sertifikat SSL, bergantung pada amannya tandatangan digital dan dapat diserang dengan menggunakan hash collision. Skenario serangan seperti ini:

1. Dodo membuat 2 dokumen A dan B, A dan B memliki hash yang identic
2. Dodo mengirim sertifikat A ke Didi yang menyetujui apa yang dikatakan sertifikat, menandatangani hashya dan mengirimkan balik ke dodo
3. Dodo mencopy tandatangan yang dikirim oleh didi
4. Dodo mengirim dokumen B ke Dudu, mengaku bahwa Didi telah menandatangani sertifikat yang berbeda.karena tandatangan digital tidak menyalahi hash, Dudu tidak dapat mengetahui bahwa itu adalah palsu.

IV. Conclusion

Kesimpulannya adalah infrastruktur kunci public dapat memiliki banyak titik serangan, dan semuanya dapat menyebabkan bocornya informasi. Serangan tersebut masih mungkin berlaku di hari sekarang walaupun beberapa telah mengalami patching/perbaikan oleh pihak yang bertanggung jawab.

References

[1] Ran Canetti: Universally Composable Signature, Certification, and Authentication. CSFW 2004, http://eprint.iacr.org/2003/239

[2] OpenSSL: Documentation ca(1)

[3] VeriSign Class definitions

[4] "Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures". *Official Journal L 013 , 19/01/2000 P. 0012 - 0020*. Annex II. Retrieved 2010-02-17.

[5] "List of certificates included by Mozilla". Mozilla.org. Retrieved 30 July 2012.[*dead link*]

[6] "DigiNotar removal by Mozilla". Mozilla.org. Retrieved 30 July 2012.

[7] "DigitNotar removal by Google". Google.com.

[8] Gambar: http://ejbca.org/older_releases/ejbca_4_0/htdocs/architecture.html

[9] http://www.secureworks.com/cyber-threat-intelligence/threats/transitive-trust/

[10] https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcRTHcP6vXm5Z1G1lnZ7DJfdf-NM5CC0zSojNmWddvlk9T0-xkwU

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Mei 2014

ttd

Muhammad Gema Akbar
13510099