STUDI MENGENAI *TIME STAMP* DAN PENGGUNAANNYA

Nicolas Andres – NIM : 13504109

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if14109@students.if.itb.ac.id

Abstrak

Dalam makalah ini dibahas mengenai studi mengenai *time stamp* dan penggunaannya dalam pengamanan suatu dokumen atau data. Pengamanan yang dimaksud adalah kewenangan sesorang yang sah atas suatu dokumen atau data. *Time stamp* merupakan segel eletronik dan juga termasuk sebuah waktu penunjuk yang diterapkan pada sebuah dokumen. Waktu penunjuk menandakan kapan dokumen diberi segel sehingga setiap dokumen memiliki waktu penunjuk yang berbeda-beda.

Time Stamp menggunakan konsep *digital signature* dan *hash function*. *Digital signature* digunakan untuk memberi segel sebuah dokumen dan dirahasiakan dengan *hash function* sehingga segel tersebut tidak dapat diubah atau dihapus oleh pihak yang tidak berkepentingan. *Hash function* yang digunakan adalah *hash function* satu arah.

Pada dasarnya, *time stamp* merupakan modifikasi dari *digital signature* dengan menggunakan *hash function*. Perbedaannya adalah penambahan waktu penunjuk dan informasi lainnya. *Hash function* yang berbeda dapat menghasilkan kekuatan segel yang berbeda pula.

Terdapat dua tipe *time stamp* yaitu menggunakan TSA (*Time-Stamping Authority*) dan *time stamp* dengan konsep *distributed trust*. Masing-masing tipe menggunakan cara yang berbeda dalam pemberian *time stamp*.

Penggunaan *time stamp* sangat penting dalam hal pendokumentasian sebuah dokumen atau data yang ingin dilindungi hak ciptanya dan mementingkan waktu pemberian segel pada dokumen tersebut. Salah satu penggunaan *time stamp* adalah penerapannya pada hasil penemuan yang belum dipublikasikan. Jika ada ilmuwan lain mempunyai penemuan yang sama maka penemuan yang diakui adalah penemuan dengan waktu pada *time stamp* yang lebih lama (dari waktu sekarang) atau lebih dahulu ditemukan.

***Kata kunci**: time stamp, digital signature, hash function*.

1 Pendahuluan

Dalam kriptografi, terdapat beberapa aspek keamaanan yaitu kerahasiaan, integritas data, otentikasi, dan nirpenyangkalan. Keempat aspek keamanan ini sebaiknya dipenuhi untuk memastikan bahwa dokumen yang diterima merupakan dokumen yang sebenarnya dikirim oleh pihak pengirim, dan bukan dokumen hasil manipulasi oleh pihak ketiga.

Aspek keamanan suatu dokumen atau pesan yang dikirim oleh satu pihak kepada pihak lainnya telah menjadi suatu hal yang kompleks dalam berbagai bidang. Dokumen yang dikirimkan dapat diubah atau dimanipulasi oleh pihak ketiga selama pengiriman. Pihak ini dapat membuat dokumen tersebut seolah-olah berasal dari pihak yang mengirim padahal dokumen tersebut telah diubah isinya sehingga dokumen tidak berisi informasi yang diharapkan oleh pengirim dan penerima.

Keamanan suatu dokumen sebenarnya dapat diatasi dengan memanfaatkan algoritma kriptografi yang telah ada misalnya algoritma kriptografi kunci publik yang memanfaatkan kunci publik dan privat pada saat melakukan enkripsi dan dekripsi. Pengirim pesan (dokumen) menggunakan algoritma tersebut untuk mengenkripsi dokumen dengan kunci publik menjadi sebuah cipherteks yang tidak dapat dimengerti oleh pihak yang tidak berwenang kecuali kunci privat yang digunakan pada saat enkripsi diketahui. Selanjutnya, pada saat cipherteks sampai pada

penerima, dokumen didekripsi dengan menggunakan kunci privat yang sesuai untuk mengetahui isi dari dokumen.

Akan tetapi, terdapat beberapa dokumen yang isi nya diperuntukan untuk dipublikasikan. Dengan memperhatikan aspek keamanan dalam kriptografi, dokumen yang dipublikasikan perlu diberi sebuah penanda bahwa dokumen tersebut "aman" dan sah. Penanda ini akan meyakinkan setiap orang bahwa dokumen tersebut memberikan informasi yang sesuai dengan keinginan pembuatnya tanpa ada perubahan terhadap isi dari dokumen tersebut.

Dengan memanfaatkan *digital signature* atau tanda tangan digital yang diberikan pada sebuah dokumen, setiap orang dapat mengetahui keabsahan suatu dokumen dengan memeriksa tanda tangan digitalnya. Apabila isi maupun tanda tangan digital diubah atau dimanipulasi oleh pihak ketiga, penerima dokumen akan mengetahui bahwa dokumen tersebut sudah diubah isinya dan tidak sah lagi (bukan dokumen asli yang dikirim). *Digital signature* ini sangat dipengaruhi dengan isi dari sebuah dokumen.

Selain itu, adakalanya sebuah dokumen memerlukan keabsahan mengenai waktu dari pemberian tanda tangan. Biasanya waktu ini digunakan sebagai waktu penunjuk kapan dokumen tersebut dibuat, sehingga, baik pembuat dokumen maupun pihak lain, tidak dapat memanipulasi waktu dari tanda tangan digital yang terdapat dalam sebuah dokumen. Penggunaan waktu ini biasanya digunakan pada sebuah dokumen laporan keuangan dalam bisnis perbankan yang rentan terhadap manipulasi isi dari dokumen dan waktu dokumen tersebut dibuat.

Metode yang dapat memecahkan persoalan keabsahan suatu dokumen, baik keaslian dokumen maupun waktu pembuatan dokumen, dan memenuhi keempat aspek keamanan dalam kriptografi adalah sebuah metode yang dinamakan *time stamp*. Setiap pembuat dokumen dapat memberikan sebuah *time stamp* sehingga setiap orang mengetahui kapan dokumen tersebut dibuat dan jika terjadi perubahan isi dokumen dapat diketahui dengan memeriksa *time stamp*-nya.

Waktu pada *time stamp* menandakan bahwa dokumen tersebut dibuat pada suatu waktu yang dapat dipertanggungjawabkan dengan menggunakan jasa pihak ketiga yaitu *time stamp authority*. Pihak ketiga diperlukan karena waktu pada setiap komputer dapat berbeda-beda. Oleh karena itu, dibutuhkan pihak ketiga yang dapat dipercaya untuk mengatasi permasalahan ini.

2 Penjelasan mengenai *Time Stamp*

Sebuah *time stamp* adalah waktu sesungguhnya dari suatu kejadian yang direkam oleh komputer dan digunakan sebagai jaminan bahwa waktu dari kejadian tersebut adalah sah atau dapat dipertanggungjawabkan oleh pemberi *time stamp*. Waktu tersebut diperlihara oleh komputer sehingga tetap akurat dengan melakukan kalibrasi hingga satuan detik.

Sebagai pengganti waktu sesungguhnya dari suatu kejadian, *time stamp* dapat berupa waktu relatif terhadap waktu lainnya. Sebagai contoh, *time stamp* dapat digunakan pada berbagai proses sinkronisassi, seperti pengurutan dari transaksi yang terjadi beberapa kali sehingga jika terjadi kesalahan pada suatu transaksi, transaksi yang sesudahnya dapat dibatalkan.

Pada dasarnya, *time stamp* menggunakan teknik *digital signature* dan *hash function* dalam membubuhi tanda tangan beserta waktu penunjuk. Oleh karena itu, pada dua bagian berikut akan dibahas sekilas mengenai *hash function* dan *digital signature*.

2.1 *Hash Function*

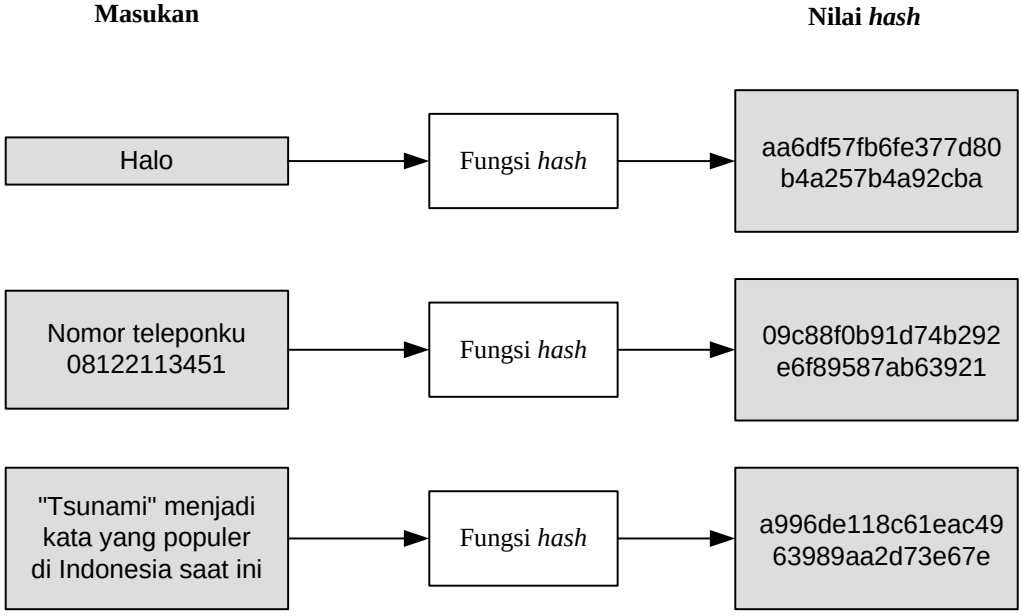
Dalam bidang kriptografi, terdapat sebuah fungsi yang sesuai untuk aplikasi keamaan seperti otentikasi dan integritas pesan. Fungsi tersebut adalah fungsi hash (kadang-kadang dinamakan juga fungsi *hash* kriptografi). Fungsi *hash* adalah fungsi yang menerima masukan *string* yang panjangnya sembarang dan mengkonversinya menjadi *string* keluaran yang panjangnya tetap (*fixed*) (umumnya berukuran jauh lebih kecil daripada ukuran *string* semula).

Fungsi *hash* dapat menerima masukan *string* apa saja. Jika *string* menyatakan pesan (*message*), maka sembarang pesan *M* berukuran bebas dikompresi oleh fungsi *hash H* melalui persamaan

$$h = H(M) \qquad (1)$$

Keluaran fungsi *hash* disebut juga nilai *hash* (*hash-value*) atau pesan-ringkas (*message digest*). Pada persamaan (1), *h* adalah nilai *hash* atau *message digest* yang ukurannya selalu tetap (dan lebih pendek dari panjang pesan semula). Gambar 1 memperlihatkan contoh 3 buah pesan dengan panjangnya tetap (dalam conoh ini pesan ringkas dinyatakan

dalam kode heksadesimal yang panjangnya 128 bit. Satu karakter heksadesimal = 4 bit). Nama lain fungsi *hash* adalah: fungsi kompresi/kontraksi (*compression function*), cetak-jari (*fingerprint*), *cryptographic checksum*, *message integrity check* (*MIC*), *manipulation deletion check* (*MDC*).



Gambar 1: Contoh *hashing* terhadap beberapa buah pesan dengan panjang berbeda-beda

Aplikasi fungsi *hash* misalnya untuk memverifikasi kesamaan salinan suatu arsip dengan arsip aslinya yang tersimpan di dalam sebuah basis data terpusat. Daripada mengirim salinan arsip tersebut secara keseluruhan ke komputer pusat (yang membutuhkan waktu transmisi lama dan ongkos yang mahal), lebih mangkus mengirimkan *message digest*-nya saja. Jika *message digest* salunan arsip sama dengan *message digest* arsip asli, berarti salinan arsip tersebut sama dengan arsip di dalam basis data.

Fungsi *hash* dapat disebut juga sebagai fungsi *hash* satu-arah. Fungsi *hash* satu-arah (*One-way Hash*) adalah fungsi *hash* yang bekerja dalam satu arah: pesan yang sudah diubah menjadi *message digest* tidak dapat dikembalikan menjadi pesan semula. Dua pesan yang berbeda akan selalu menghasilkan nilai hash yang berbeda pula. Sifat-sifat fungsi *hash* satu-arah adalah sebagai berikut:

1. Fungsi H dapat diterapkan pada blok data berukuran berapa saja.
2. H menghasilkan nilai (h) dengan panjang tetap (*fixed-length output*).
3. $H(x)$ mudah dihitung untuk nilai x yang diberikan.
4. Untuk setiap h yang diberikan, tidak mungkin menemukan x sedemikian sehingga $H(x) = h$. Itulah sebabnya fungsi H dikatakan fungsi *hash* satu-arah (*one-way function*).
5. Untuk setiap x yang diberikan, tidak mungkin mencari $y \neq x$ sedemikian sehingga $H(y) = H(x)$.
6. Tidak mungkin (secara komputasi) mencari pasangan x dan y sedemikian sehingga $H(x) = H(y)$.

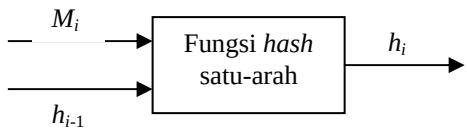
Keenam sifat di atas penting sebab fungsi *hash* seharusnya berlaku seperti fungsi acak. Sebuah fungsi *hash* dianggap tidak aman jika (i) secara komputasi dimungkinkan menemukan pesan yang bersesuaian dengan pesan ringkasnya, dan (ii) terjadi kolisi (*collision*), yaitu terdapat beberapa pesan berbeda yang mempunyai pesan ringkas yang sama.

Fungsi *hash* bekerja secara iteratif. Masukan fungsi *hash* adalah blok pesan (M) dan keluaran dari *hashing* blok pesan sebelumnya,

$$h_i = H(M_i, h_{i-1}) \qquad (2)$$

Skema fungsi *hash* ditunjukkan pada gambar 2. Fungsi *hash* adalah publik (tidak dirahasiakan), dan keamanannya terletak pada sifat satu arahnya itu. Ada beberapa fungsi *hash* satu-arah yang dibuat orang, antara lain: *MD2*, *MD4*, *MD5*, *Secure Hash Function* (*SHA*), *RIPMEND*, *WHIRPOOL*, dan lain-lain.

Tabel 1 meresumekan parameter beberapa fungsi *hash*.



Gambar 2: Fungsi *hash* satu-arah

Penggunaan fungsi *hash* satu-arah lebih banyak diterapkan pada penyimpanan *password* (sandi lewat) yang membutuhkan keamaan. Pihak-pihak yang mempunyai hak akses terhadap basis data tempat penyimpanan *password*, tidak dapat mengetahui *password* sebenarnya karena yang tersimpan adalah *message digest* dari *password*. Selain itu, sifat dari fungsi *hash* satu-arah tidak memungkinkan untuk mengetahui pesan yang di-*hashing* dengan hanya mengetahui *message digest* dari pesan tersebut.

Dalam hal pencocokan *password* yang dimasukkan, *password* akan dihitung terlebih dahulu *message digest*-nya. Kemudian, *password* hasil *hashing* (fungsi *hash*) akan dicocokan dengan *message digest* yang tersimpan dalam basis data. *Password* akan dianggap sah atau valid jika *message digest* hasil perhitungan sama dengan *message digest* yang tersimpan dalam basis data.

Tabel 1 Beberapa fungsi *hash*

Algoritma	**Ukuran *message digest* (bit)**	**Ukuran blok pesan**	**Kolisi**
MD2	128	128	Ya
MD4	128	512	Hampir
MD5	128	512	Ya
RIPEMD	128	512	Ya
RIPEMD-128/256	128/256	512	Tidak
RIPEMD-160/320	160/320	512	Tidak
SHA-0	160	512	Ya
SHA-1	160	512	Ada cacat
SHA-256/224	256/224	512	Tidak
SHA-512/384	512/384	1024	Tidak
WHIRLPOOL	512	512	Tidak

2.2 *Digital Signature*

Sudah sejak lama, tanda tangan telah menjadi suatu penanda sebuah dokumen. Tanda tangan digunakan untuk membuktikan otentikasi dokumen kertas (misalnya surat, piagam, ijazah, buku, karya seni, laporan keuangan, dan sebagainya). Tanda tangan mempunyai karakteristik sebagai berikut:

1. Tanda tangan adalah bukti yang otentik.
2. Tanda tangan tidak dapat dilupakan.
3. Tanda tangan tidak dapat dipindah untuk digunakan ulang.
4. Dokumen yang telah ditandatangani tidak dapat diubah.
5. Tanda tangan tidak dapat disangkal (*repudiation*).

Fungsi tanda tangan pada dokumen kertas juga diterapkan untuk otentikasi pada data digital seperti pesan yang dikirm melalui saluran telekomunikasi dan dokumen elektronis yang disimpan di dalam memori komputer. Tanda tangan pada data digital ini dinamakan tanda-tangan digital (*digital signature*). Yang dimaksud dengan tanda-tangan digital bukanlah tanda tangan yang digitisasi dengan alat *scanner*, tetapi suatu nilai kriptografis yang bergantung pada pesan dan pengirim pesan (Hal ini kontras dengan tanda tangan pada dokumen kertas yang bergantung hanya pada pengirim dan selalu sama untuk semua dokumen). Dengan tanda-tangan digital, maka integritas data dapat dijamin, disamping itu dapat juga digunakan untuk membuktikan asal pesan (keabsahan pengirim), dan nirpenyangkalan.

Menandatangani pesan dapat dilakukan dengan salah satu dari dua cara berikut:

1. Enkripsi pesan
   Mengenkripsi pesan dengan sendirinya juga menyediakan ukuran otentikasi. Pesan yang terenkripsi sudah menyatakan bahwa pesan tersebut telah ditandatangani.
2. Tanda-tangan digital dengan fungsi *hash* (*hash function*)
   Tanda-tangan digital dibangkitkan dari *hashing* terhadap pesan. Nilai *hash* adalah kode ringkas dari pesan. Tanda-tangan digitak ditambahkan (*append*) pada pesan.

Penandatangan pesan dengan cara mengenkripsinya selaku memberikan dua fungsi berbeda: kerahasiaan pesan dan otentiksasi. Pada beberapa kasus, seringkali otentikasi yang diperlukan, tetapi kerahasiaan pesan tidak. Maksudnya, pesan tidak perlu dienkripsikan, sebab yang dibutuhkan hanya otentikasi saja.

Hanya sistem kriptografi kunci-publik yang cocok dan alami untuk pemberian tanda-tangan digital dengan menggunakan fungsi *hash*. Hal ini disebabkan masalah *non-repudiation* (baik penerima dan pengirim pesan mempunyai pasangan kunci masing-masing).

Proses pemberian tanda-rangan digital (*signing*) dimulai dengan menghitung *message digest* dari pesan. *Message digest* (*MD*) diperoleh dengan mentransformasikan pesan *M* dengan menggunakn fungsi *hash* satu-arah *H*,

$$MD = H(M) \qquad (3)$$

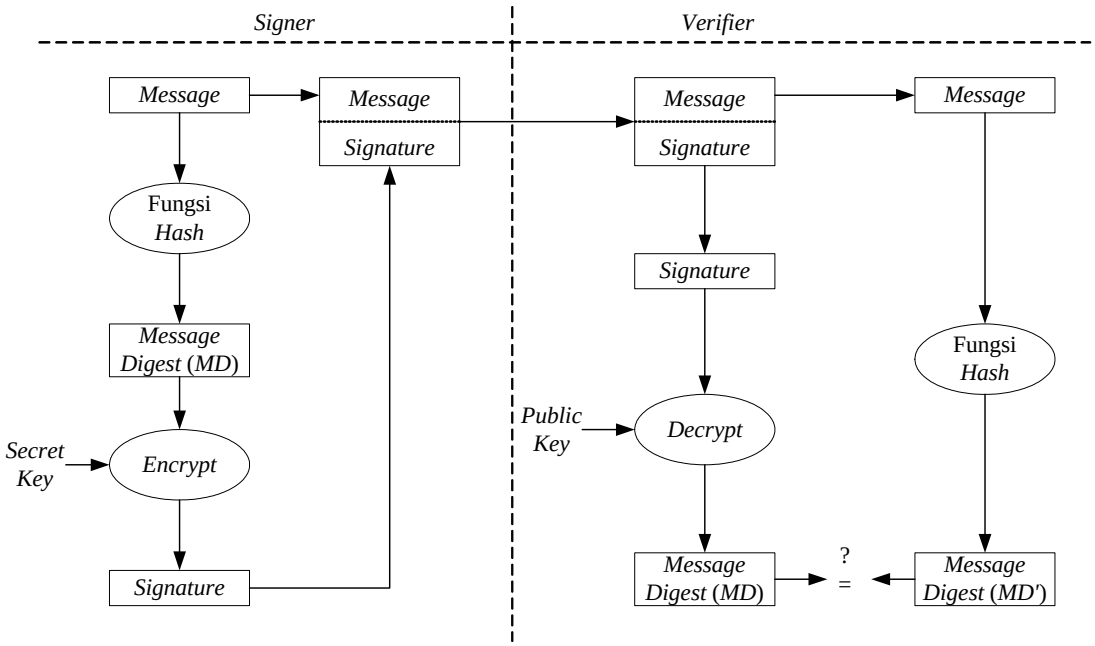
Selanjutnya, *message digest MD* dienkripsi dengan algoritma kriptografi kunci-publik dan menggunakan kunci privat (*SK*) si pengirim. Hasil enkripsi inilah yang dinamakan dengan tanda-tangan digital *S*,

$$S = E_{SK}(MD) \qquad (4)$$

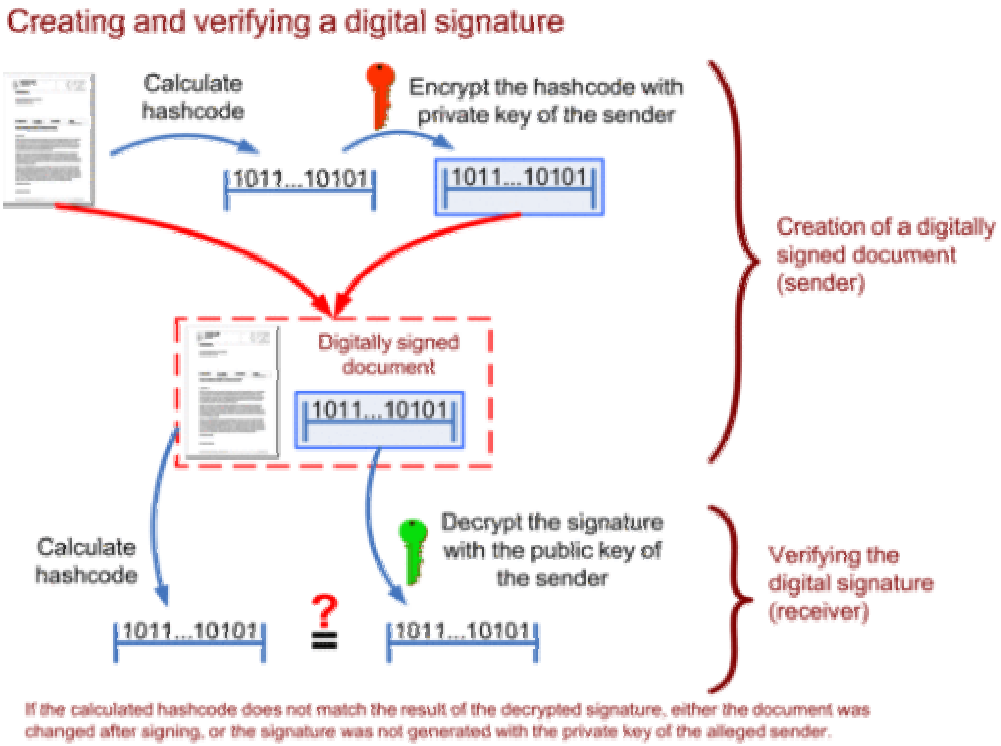
Kemudian, tanda-tangan digital *S* dilekatkan ke pesan *M* (dengan cara menyambung/*append*) *S*, lalu keduanya dikirim melalui saluran komunikasi. Dalam hal ini, pesan *M* dikatakan sudah ditandatangani oleh pengirim dengan tanda-tangan digital *S*.

Ketika pesan yang telah ditandatangani sampai ke penerima, tanda-tangan diverifikasi untuk dibuktikan keotentikannya dengan cara berikut:

1. Tanda-tangan digital *S* didekripsi dengan menggunakan kunci publik (*PK*) pengirim pesan, sehingga menghasilkan *message digest* semula, *MD*, sebagai berikut:
   $$MD = D_{PK}(S) \qquad (5)$$
2. Penerima kemudian mengubah pesan *M* menjadi *message digest MD'* menggunakan fungsi *hash* satu-arah yang sama dengan fungsi *hash* yang digunakan oleh pengirim.
3. Jika *MD' = MD*, berarti tanda-tangan yang diterima otentik dan berasal dari pengirim yang benar.



Gambar3: Otentikasi dengan tanda-tangan digital yang menggunakan fungsi *hash* satu-arah



Gambar 4: Visualisasi pemberian tanda-tangan digital dan verifikasinya

Skema tanda-tangan digital yang menggunakan fungsi *hash* ditunjukkan pada gambar 3. Gambar 4 memperlihatkan visualisasi penandatanganan dan pemverifikasian tanda-tangan.

Otentikasi pesan dijelaskan sebagai berikut:

a. Apabilia pesan *M* yang diterima sudah berubah, maka *MD* yang dihasilkan dari fungsi *hash* berbeda dengan *MD* semula yang didapatkan dengan mendekripsi tanda-tangan digital menggunakan kunci publik pengirim pesan. Ini berarti pesan tidak asli lagi atau sudah berubah.
b. Apabila pesan *M* tidak berasal dari orang yang sebenarnya, maka *message digest MD* yang dihasilkan dari 3 persamaan (persamaan 3, 4, dan 5), akan berbeda dengan *message digest MD*' yang dihasilkan pada saat proses verifikasi (hal ini karena kunci publik yang digunakan oleh penerima pesan tidak berkoresponden dengan kunci privat pengirim).
c. Bila *MD* = *MD*', ini berarti pesan yang diterima adalah pesan yang asli (*message authentication*) dan orang yang mengirim pesan adalah orang yang sebenarnya (*user authenticationi).*

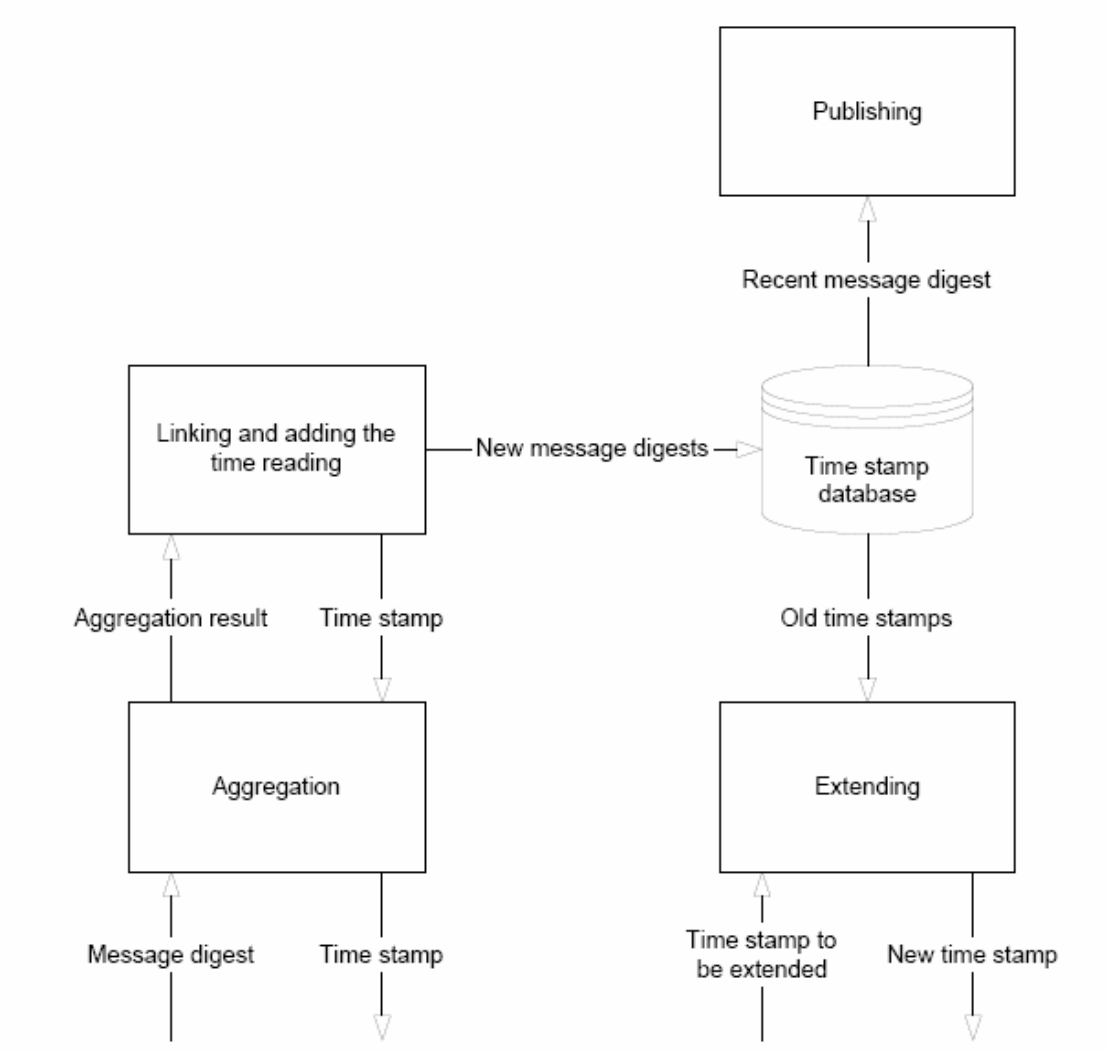
Pengirim pesan tidak dapat menyangkal pesan yang dikirim, sebab tanda-tangan digital dapat digunakan untuk melakukan nirpenyangkalan. Andaikan pengirim berbohong telah mengirim pesan. Sangkalan dari pengirim dapat dibantah dengan cara sebagai berikut: jika ia tidak mengirim pesan, berarti ua tidak menenkripsi *message digest* dari pesan dengan kunci privatnya. Faktanya, kunci publik yang berkoresponden dengan kunci privat pengirim menghasilkan *MD = MD'*. Ini berarti *message digest* memang benar dienkripsi oleh pengirim sebab hanya yang mengetahui kunci privatnya sendiri.

Dua algoritma *signature* yang digunakan secara luas adalah *RSA* dan *El Gamal*. Pada *RSA*, algoritma enkripsi dan dekripsi identik, sehingga proses *signature* dan verifikasi juga identik. Selain *RSA*, terdapat algoritma lain yang dikhususkan untuk tanda-tangan digital, yaitu *Digital Signature Algorithm* (*DSA*), yang merupakan bakuan (*standard*) untuk *Digital Signature Standard* (*DSS*). Pada *DSA*, algoritma *signature* dan verifikasi berbeda.

3 Struktur umum sebuah sistem *Time-Stamping*

Time-Stamping Service pada umumnya terdiri dari lima kompenen dengan fungsionalitas yang berbeda-beda (lihat gambar 5).

1. *Aggregation* (gambar 6) – sebuah komponen yang menerima sebuah rangkaian permintaan *time-stamping* dan membentuk suatu keluaran permintaan yang disebut *authentication tree*. Tujuan dari *aggregation* adalah meningkatkan performansi sistem dengan mengurangi pekerjaan *server* sehingga jumlah permintaan yang dikirimkan secara langsung ke *server* dapat dikurangi. Bagian *aggregation* tidak harus ada jika *server* mempunyai kekuatan yang besar untuk menangani semua permintaan. Di lain pihak, komponen ini tidak menimbulkan resiko keamanan akan *availability* (misalnya resiko hilangnya permintaan yang lebih besar). Oleh karena itu, tidak diperlukan pembuktian secara terpisah bahwa komponen *aggregation* adalah benar.

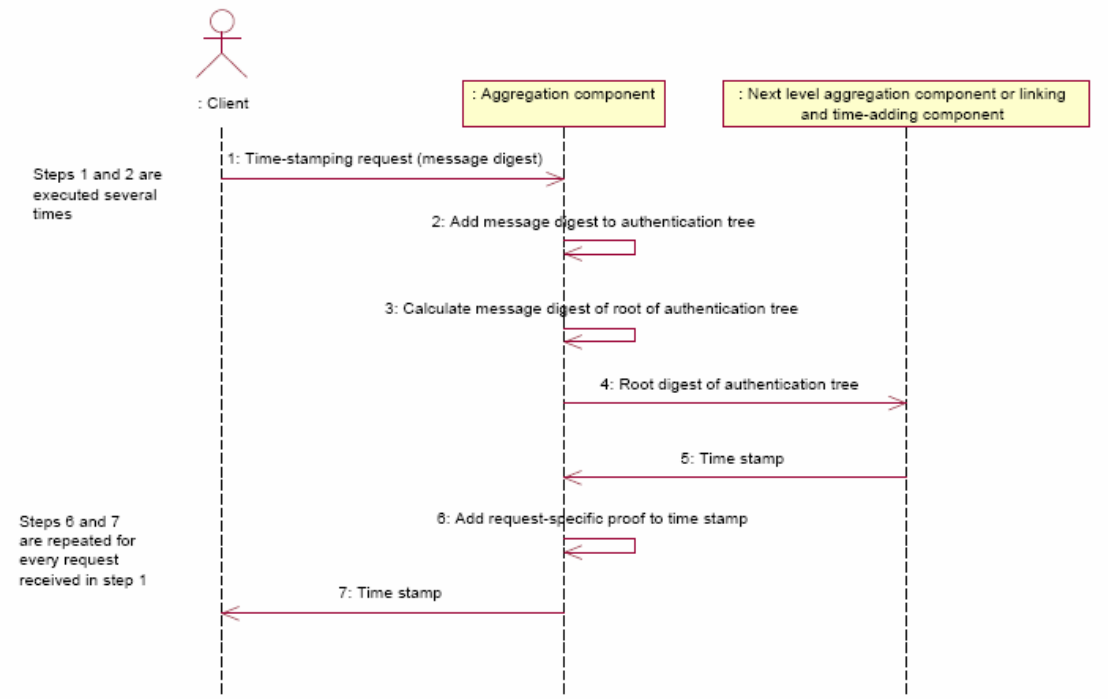


Gambar 5: Struktur umum dari *Time-Stamping Service*

2. *Adding the time value* – sebuah komponen yang menambahkan waktu pada *time stamp* yang akan dibuat.
3. *Linking* – sebuah komponen yang membuat sebuah cara yang menghubungkan antara *time stamp* yang baru dengan semua atau beberapa *time stamp* yang lama, dan memungkinkan perbandingan secara langsung serta memeriksa *Time-Stamping Service*. Keterhubungan tersebut dibuat dengan menghitung nilai *hash* (*message digest*) menggunakan *hash function*. *Time*

stamp yang baru juga ditandatangani oleh *server* untuk menjamin keabsahan dan pembuktian kebenaran secara cepat.

4. *Publishing* – komponen ini digunakan untuk membuat sebuah perhitungan dari nilai *hash* terakhir yang tersedia oleh beberapa media umum dan yang memeriksa kebenaran dari *time stamp*, misalnya dengan mencetaknya dalam koran. *Publishing* adalah komponen penting untuk meyakinkan dapat dibuktikannya kebenaran dari *service* dan memelihara kebenaran dari *time stamp* dalam kurun waktu yang lama.



Gambar 6: Cara kerja komponen *Aggregation*

5. *Extending* – komponen ini digunakan untuk memperbaharui informasi yang dibutuhkan untuk pemeriksaan kebenaran dari *time stamp*. Ketika melakukan *extending* pada sebuah *time stamp*, *server* membuat sebuah rantai *hash* yang membuktikan bahwa *time stamp* yang akan dilakukan *extending* ada sebelum permintaan *time stamp* lain oleh *client*. *Extending* yang paling berguna adalah *extending* terhadap *time stamp* terakhir yang dipublikasikan. Hal ini memungkinkan pembuktian kebenaran dari *time stamp* tanpa adanya campur tangan atau gangguan dari *Time-Stamping Server*.

4 Dua Teknik *Time Stamp*

Ide mengenai *time stamp* sebagai waktu penunjuk yang diberikan pada sebuah dokumen digital telah digunakan beberapa abad yang lalu. Sebagai contoh, ketika Robert Hooke menemukan Hukum Hooke pada tahun 1660, dia tidak ingin penemuannya tersebut dipublikasikan, tetapi dia ingin dianggap sebagai penemu yang pertama. Sehingga dia mempublikasikan *anagram ceiiinosssttuv* dan kemudian mempulikasikan terjemahannya *ut tensio sic vis*. Dalam kehidupan modern, contoh kasus penggunaan *time stamp* adalah dalam organisasi penelitian dimana suatu penemuan tertentu ditemukan pada suatu waktu dan membutuhkan sebuah pengakuan (*patent*) sehingga tidak ada peneliti lain yang dapat mengakui penelitian orang lain sebagai penelitiannya.

Terdapat dua tipe dari teknik *time stamp* yaitu *time stamp* yang bekerja dengan sebuah *Trusted Third Party* (*Time-Stamping Authority - TSA*) dan *time stamp* dengan konsep dari *distributed trust*. Teknik yang berdasarkan kepercayaan pada *TSA* adalah pihak yang dipercaya dalam memberi *Digital Time Stamp*. Sedangkan teknik yang berdasarkan pada *distributed trust* terdiri dari tanda tangan dan

waktu dari dokumen yang dilakukan oleh beberapa orang (*client*) dari sebuah kelompok untuk meyakinkan bahwa tidak ada orang (*client*) yang dapat mengubah elemen-elemen dari *time stamp*.

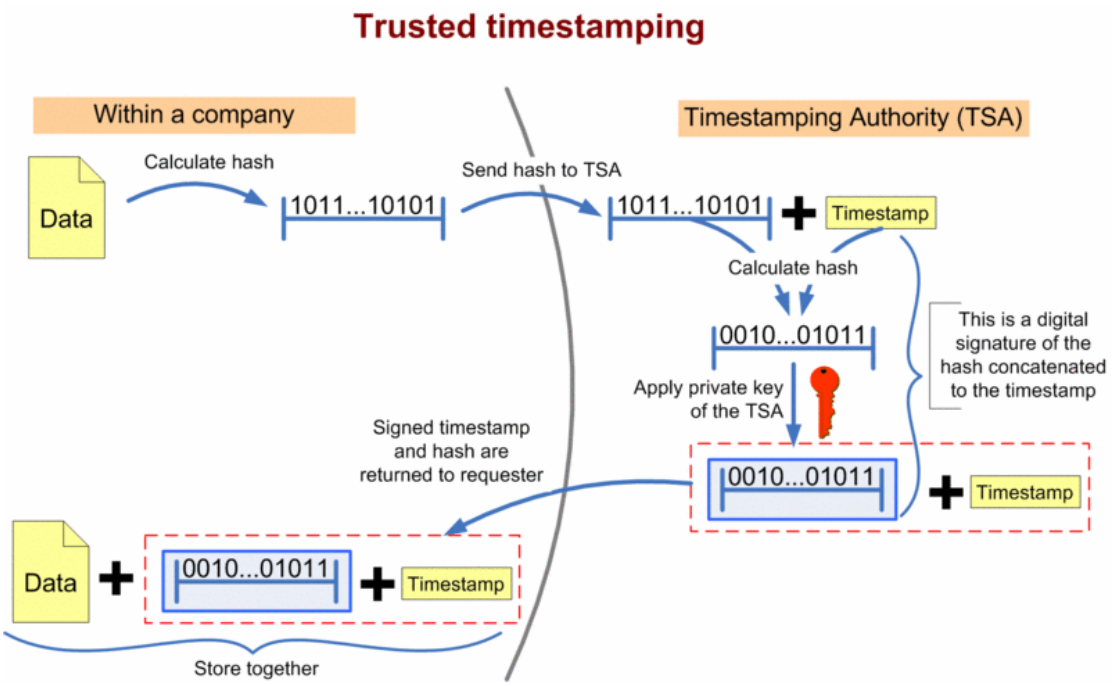
4.1 *Time Stamp* dengan *TSA*

Time-Stamping authority (*TSA*) bertindak sebagai pihak yang dipercaya oleh pihak yang ingin memberikan *time stamp* pada sebuah dokumen. Pihak ini berperan dalam membuktikan keberadaan sebuah dokumen tanpa ada kemungkinan pihak pemilik dokumen dapat mengubah waktu penunjuk pada *time stamp* sehingga penerima dokumen dapat yakin dengan *time stamp* yang ada dalam dokumen tersebut.

Pemberian *time stamp* dengan menggunakan jasa *TSA* biasanya dinamakan dengan *Tursted Digital Timestamping* karena biasanya pihak *TSA* telah dipercaya oleh orang yang ingin memberikan *time stamp* pada dokumennya. *Time stamp* diberikan pada dokumen dengan cara mengirimkan dokumen yang akan diberi *time stamp* kepada *TSA* dan kemudian *TSA* mengirimkan kembali dokumen yang telah diberi *time stamp* kepada pengirim dokumen asli.

Trusted timestamping adalah sebuah proses yang menjamin keamanan waktu pembuatan dan pengubahan sebuah dokumen. Keamanan disini berarti bahwa tidak ada satupun, bahkan pemilik dokumen sendiri, yang dapat mengubah *time stamp* dari dokumen sejak *time stamp* telah diberikan oleh *TSA*.

Trusted timestampin digunakan untuk membuktikan keberadaan sebuah dokumen tertentu (contohnya surat kontrak, data penelitian, catatan medis) tanpa ada kemungkinan pemilik dokumen dapat mengubah waktu dari *time stamp*. *TSA* yang berbeda dapat digunakan untuk meningkatkan *reliability* dan mengurangi *vulnerability*.



Gambar 7: Mendapatkan *trusted time stamp* dari *TSA*

Pembuatan sebuah *time stamp* didasarkan pada *digital signature* dan *hash function*. Pertama, sebuah nilai *hash* dihitung dari dokumen. Nilai *hash* dapat dikatakan sebagai *digital fingerprint* dari dokumen asli dan merupakan sebuah *string* yang berbeda untuk setiap dokumen. Jika dokumen asli diubah maka nilai *hash* akan menghasilkan *string* yang berbeda dari dokumen asli. Nilai *hash* ini dikirimkan ke *TSA*. *TSA* kemudian menggabungkan sebuah *time stamp* dengan nilai *hash* dan menghitung nilai *hash* yang baru dari hasil penggabungan. Nilai *hash* yang baru ini ditandatangani dengan menggunakan kunci privat dari *TSA*. Nilai *hash* dari dokumen asli dan *time stamp* yang telah ditandatangani dikirimkan kembali ke

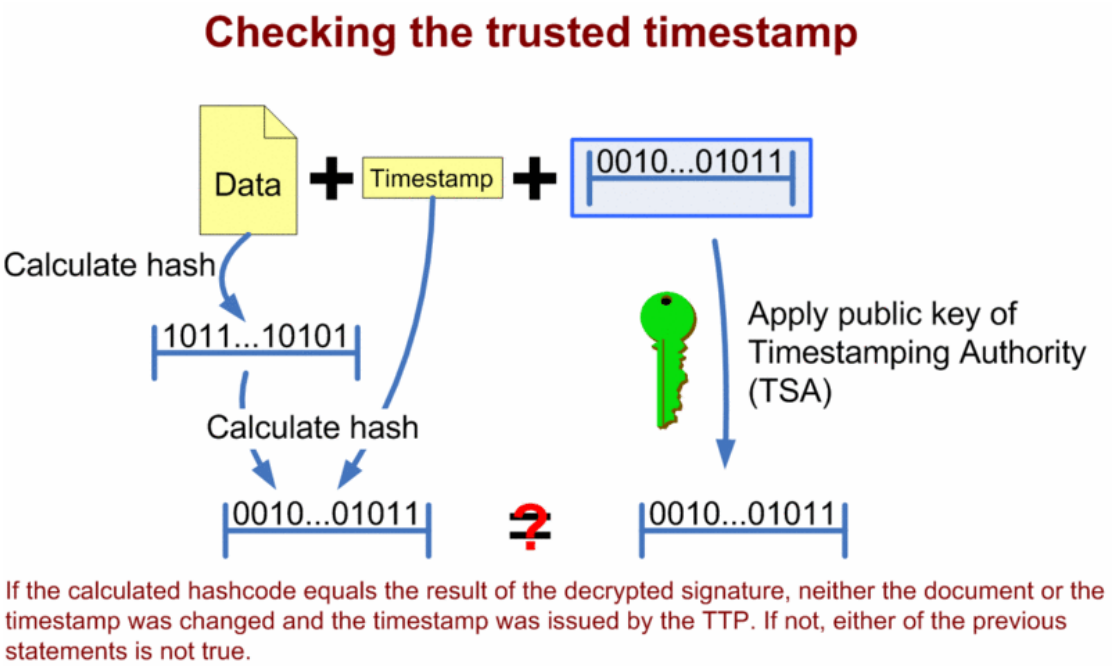
pengirim dokumen asli dan menyimpannya bersama dengan dokumen asli. (lihat gambar 7). Semenjak dokumen asli tidak bisa didapatkan dari nilai *hash*-nya (karena *hash function* bersifat satu arah), *TSA* tidak pernah dapat melihat dokumen asli sehingga memungkinkan penggunaannya untuk dokumen rahasia.

Siapapun yang mempercayai sebuah *TSA* dapat mengecek kebeneran bahwa dokumen tidak dibuat setelah tanggal saat *TSA* menjamin tanggal dokumen dibuat. Selain itu, dapat juga diperiksa bahwa tidak ada penyangkalan mengenai pengirim dokumen yang meminta *time stamp* merupakan pemilik sebenarnya dari dokumen asli ketika *time stamp* diberikan. Untuk membuktikannya (lihat gambar 8), nilai *hash* dari dokumen asli dihitung terlebih dahulu, kemudian *time stamp* yang diberikan oleh *TSA* digabungkan dan hitung nilai *hash* dari hasil penggabungan ini, misalnya *hash* A.

Setelah itu, *digital signature* dari *TSA* diperiksa dengan mendekripsi nilai *hash* yang telah ditandatangani oleh *TSA* dengan menggunakan kunci publik dari *TSA*. Hasil dekripsi ini misalnya dinamakan *hash* B. Jika *hash* A sama dengan *hash* B maka *time stamp* atau dokumen tidak pernah diubah dan benar dikeluarkan oleh *TSA* yang digunakan saat pemberian *time stamp*.

4.2 *Time Stamp* dengan *Distributed Trust*

Untuk tipe ini, dapat diasumsikan bahwa terdapat sebuah skema tanda tangan yang aman sehingga setiap *user* dapat menandatangani pesan, dan tersedia sebuah standar *pseudorandom generator G* yang aman untuk semua *user*. Sebuah *pseudorandom generator* adalah sebuah algoritma yang menerima masukkan berupa nilai awalan (*seed*) yang pendek dan menghasilkan rangkaian nilai keluaran yang tidak dapat dibedakan oleh algoritma dari rangkaian yang acak, dengan kata lain, rangkaian keluaran tidak dapat diperkirakan.



Gambar 8: Mengecek kebenaran *time stamp* yang dibuat oleh *TSA*

Selanjutnya, misalkan sebuah nilai *hash* y yang akan digunakan oleh *client* (orang yang akan memberi *time stamp* sebuah dokumen) kita dalam pemberian *time stamp*. y digunakan sebagai nilai awalan (*seed*) untuk *pseudorandom generator*, dan memiliki keluaran yang dapat diinterpretasikan dalam sebuah cara standar sebagai sebuah *k-tuple* dari nomor-nomor pengenal *client*:

$$G(y) = (ID_1, ID_2, ..., ID_k)$$

Permintaan untuk melakukan *time stamp* (*y*, ID) dikirimkan kepada setiap *client*. Hasil permintaan akan berupa pesan yang telah ditandatangani oleh *client* ID_j. Pesan tersebut adalah $sj = \sigma_j(t, ID, y)$ dimana *t* adalah waktu. *Time stamp* akan terdiri dari [(*y*, ID);

(s_1,....,s_k)]. *k* tanda tangan dapat dengan mudah dicek oleh pihak yang meminta *time stamp*.

List dari tanda tangan dapat menjadi sebuah *time stamp* yang dapat dipercaya karena dalam lingkupnya, satu-satunya cara untuk membuat sebuah dokumen yang telah diberi *time stamp* dengan waktu yang salah adalah menggunakan sebuah nilai *hash y* dimana *G*(*y*) menghasilkan *k client* yang bekerja sama dalam membuat *time stamp* yang salah. Jika suatu saat terdapat sebuah *fraction* yang paling konstan ε dari kemungkinan *client* yang tidak jujur, jumlah dari nilai awalan (*seed*) *y* yang diharapkan dan harus dicoba sebelum menemukan sebuah *k-tuple G*(*y*) yang hanya mengandung *client* lain yang diajak kerja sama dari antara *fraction* ini adalah ε^{-k}. Selanjutnya, sejak *G* diasumsikan sebagai sebuah *pseudorandom generator* yang aman, tidak terdapat cara lain yang lebih cepat dalam menemukan nilai awalan (*seed*) *y* yang cocok dibandingkan dengan memilihnya secara acak.

Parameter *k* sebaiknya dipilih ketika merancang sistem sehingga ini merupakan perhitungan yang tidak mungkin. Dengan melihat adanya kemungkinan sebuah keraguan yang besar dalam memperkirakan persentase dari *client* yang mungkin dapat disuap (diajak kerja sama dalam menghasilkan *time stamp* yang salah) -ε mungkin 90%- tidak memerlukan sebuah pemilihan *k* yang besar dan dapat mencegah hal tersebut. Sebagai tambahan, *list* dari *client* yang dapat disuap tidak perlu diperbaiki, sepanjang *fraction* mereka dalam populasi tidak tidak melebihi ε.

Tipe ini tidak pernah menggunakan sebuah *Time-Stamping Service* yang terpusat. Hanya terdapat persyaratan bahwa memungkinkan untuk memanggil *client* lain dan menerima tanda tangan yang diinginkan dari mereka, serta terdapat sebuah direktori umum dari *client* sehingga memungkinkan untuk menginterpretasi keluaran dari *G*(*y*) dalam sebuah cara standar sebagai sebuah *k-tuple* dari *client*. Sebuah implementasi praktis dari tipe ini mungkin membutuhkan ketentuan dalam protokol bagi *client* yang tidak dapat diubah pada saat permintaan *time-stamping*. Sebagai contoh, untuk *k'* < *k* yang cocok, sistem mungkin menerima respon dari beberapa *k'* dalam *k clients* yang digunakan oleh *G*(*y*) sebagai sebuah *time-stamp* yang sah untuk *y* (dalam kasus sebuah nilai yang lebih besar untuk parameter *k* mungkin dibutuhkan untuk memperkecil kemungkinan dalam menemukan sebuah kumpulan *client* yang bekerja sama secara acak).

5 *Time Stamp Scheme*

Dalam bagian ini akan dibahas beberapa *scheme* yang digunakan dalam *time stamp*. *Scheme-scheme* ini digunakan dalam tipe *time stamp* menggunakan pihak lain yaitu *TSA* yang menyediakan jasa *TSS* (*Time-Stamping Service*). Masing-masing *scheme* menggunakan cara yang berbeda satu sama lain dalam pemberian *time stamp* pada dokumen-dokumen.

Ketika seorang *customer* mempunyai sebuah dokumen untuk diberi *time stamp*, dia mengirimkan nilai *hash* dari dokumen tersebut ke *Time-Stamping Service – TSS*. Kemudian *TSS* melampirkan tanggal dan waktu ke nilai *hash*, menandatanganinya dan mengirimkan hasil penandatanganan ke *customer*. *Customer* memeriksa tanda tangan dan menjamin bahwa nilai *hash* dari hasil dekripsi tanda tangan yang dikirimkan kepadanya merupakan nilai *hash* yang sama dengan yang dikirimkan ke *Time-Stamping Service*.

Terdapat sebuah pertanyaan yaitu bagaimana dokumen menerima *time stamp* (data dan waktu otentikasi). Ada dua cara dalam pemberian *time stamp* pada sebuah dokumen: *absolute* dan *relative*. *Absolute authentication* mengandung informasi dari tanggal dan waktu sesungguhnya yang sama dengan yang digunakan dalam dunia nyata, sementara *relative authentication* mengandung informasi yang hanya memeriksa apakah sebuah dokumen telah diberi *time stamp* sebelum atau sesudah dokumen yang lain.

Dua pola *authentication* tersebut dapat digunakan untuk memberi *time-stamp* suatu dokumen, tapi pola *absolute* mengisyaratkan bahwa *Time-Stamping Service* merupakan pihak yang dapat dipercaya. Pada pola *relative*, keberadaan pihak yang dapat dipercaya tidak dibutuhkan, oleh karena itu, terdapat beberapa mekanisme yang menjamin bahwa sebuah dokumen akan selalu mempunyai *time stamp* dengan tanggal dan waktu sekarang meskipun *Time-Stamping Service* merupakan pihak yang bermaksud jahat. Mekanisme tersebut dijelaskan sebagai berikut dan bagaimana cara kerjanya dengan pola *relative authentication*.

5.1 *Linking Scheme*

Untuk menjamin bahwa *TSA* akan memberikan *time stamp* dengan tanggal dan waktu yang berbeda untuk setiap dokumen, *Time-Stamping Service* dapat menghubungkan semua nilai *hash* dari dokumen yang telah diajukan dalam sebuah rantai menggunakan sebuah *hash*

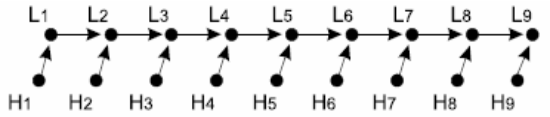
function H. Dalam kasus ini, *time stamp s* untuk dokumen ke-n H_n akan menjadi:

$$s = Sig_{TSA}(n; t_n; ID_n; H_n; L_n) \qquad (1)$$

dimana Sig_{TSA} adalah *digital signature* dari *Time-Stamping Service*, t_n adalah tanggal dari dokumen, ID_n adalah tanda pengenal dari dokumen, dan L_n adalah *link information*:

$$L_n = (t_{n-1}; ID_{n-1}; H_{n-1}, H(L_{n-1})) \qquad (2)$$

dimana t_{n-1} dan ID_{n-1} adalah tanggal dan tanda pengenal dari dokumen sebelumnya, H_{n-1} adalah nilai *hash* dari dokumen sebelumnya, dan $H(L_{n-1})$ adalah nilai *hash* dari *link information* pada dokumen sebelumnya. Gambar 9 menunjukkan bagaimana *linking* dari 9 elemen.



Gambar 9: *Linking scheme*

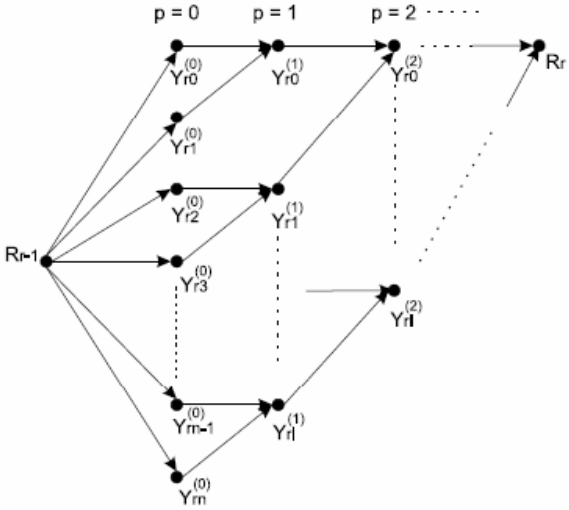
Dengan cara ini, *Time-Stamping Service* menghubungkan *time stamp* sekarang dengan yang sebelumnya untuk mendapatkan sebuah rangkaian *time stamp* dengan urutan yang masih dalam batas. Untuk memecahkan masalah perselisihan yang mungkin timbul, *Time-Stamping Service* dapat mengenali jika sebuah dokumen yang telah diberi *time stamp* sebelum yang lainnya.

Akan tetapi, terdapat beberapa masalah dengan implementasi dari cara ini. Salah satunya mungkin menghabiskan waktu dalam memeriksa hubungan antara dua *time stamp*, ini secara langsung sebanding dengan jumlah dari *time stamp* yang terhubung dalam rantai. Masalah lain yang mungkin adalah pertanyaan mengenai ketahanan dari *Time-Stamping Service*, misalkan ada dua dokumen yang datang pada waktu t1 dan t2, *Time-Stamping Service* tidak boleh membuat *time stamp* untuk dokumen yang datang pada t2 sebelum membuat *time stamp* untuk dokumen yang datang pada t1.

5.2 *Tree Scheme*

Untuk memecahkan masalah diatas, sebuah generalisasi dari *linking scheme* dibentuk, dinamakan *tree scheme*. Dasar pemikiran dari *scheme* ini adalah membagi rantai dalam unit-unit yang dinamakan *round*. Dalam setiap *bound*, seseorang dapat mengajukan beberapa daftar permintaan *time stamp* ke *Time-Stamping Service*, dan pada akhirnya *Time-Stamping Service* menghasilkan sebuah *time stamp* yang merepresetasikan semua permintaan dalam *round* tersebut. Ini terjadi sepanjang pembentukan dari *tree*, dimana daun-daun dari *tree* merupakan daftar permintaan *time stamp* dan *node*-nya dihitung menggunakan sebuah fungsi *F*, yang dapat didefinisikan sebagai sebuah *hash fucntion*, sebuah xor atau sebuah operasi yang tidak memungkinkan terjadinya *collision*.

Setiap *customer* yang ingin memberi *time stamp* minimal pada sebuah dokumen dalam sebuah *round r*, mengirimkan nilai *hash* dari dokumen tersebut ke *Time-Stamping Service*, yang didefinisikan sebagai y_{ri}. Daun dari *tree* dibentuk oleh y_{ri} yang berbeda. Setiap node *k* dari *tree* dinamakan sebagai $F_k = (F_{kL}, F_{kR})$, dimana *kL* dan *kR* adalah kiri dan kanan dari *node k*.



Gambar 10: *Tree scheme*

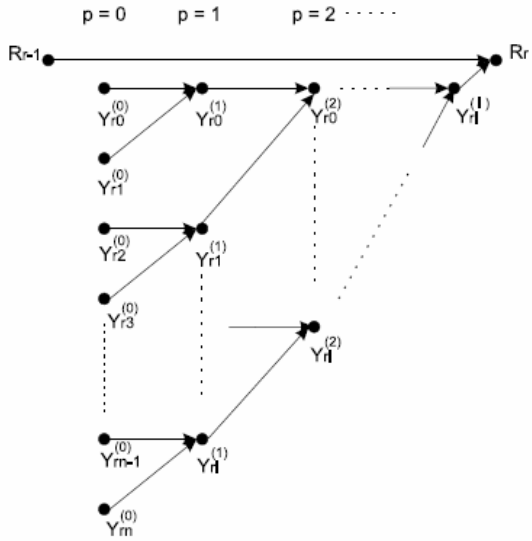
Gambar 10 menggambarkan *scheme* yang dijabarkan diatas, yaitu titik awal untuk pembentukan *tree* merupakan nilai-nilai *hash* yang diajukan ke *Time-Stamping Service* dalam *round* itu, dengan nilai-nilai *hash* ini nilai-nilai dari setiap *node* dihitung dengan menggunakan *function F* terhadap daun-daun y_{ri} dua per dua, seperti yang telihat pada gambar 10. Dimana *p* adalah indeks *level* dari *tree*, y_{ri} adalah masing-masing nilai dari node yang diturunkan dari *function F*:

$$y_{ri}^{(p)} = F(y_{r(2i)}^{(p-1)}, y_{r(2i+1)}^{(p-1)}) \qquad (3)$$

dan *l* adalah indeks maksimum dari y_{ri} yang terdapat dalam setiap *level* dan didefinsikan sebagi berikut:

$$l = [(n+1)div(2p)] - 1 \qquad (4)$$

Sebagai contoh, sebuah nilai *hash* tertentu y_{r3} didefinisikan sebagai [*r*; (y_{r4}, *L*), (F_4, *R*)] dan pembuktiannya adalah $R_r = F(F(F_4, F(y_{r3}, y_{r4})), R_{r-1})$. Disini dapat dilihat bahwa waktu yang dihabiskan untuk pemeriksaan adalah perhitungan logaritmik, sejak pencarian sebuah *time stamp* dibuat dalam sebuah rentang waktu (periode) dan tidak dalam seluruh rantai, seperti pada *linking scheme*.



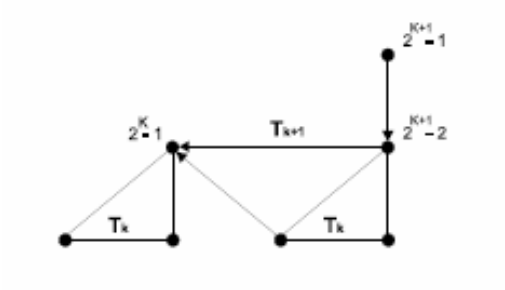
Gambar 11: Contoh lain dari sebuah *Tree scheme*

Teknik ini juga memungkinkan adanya variasi, karena *time stamp* sebelumnya R_{r-1} dapat secara langsung dihubungkan dengan *time stamp* sekarang R_r, dimana memberikan keuntungan untuk proses pemberian *time stamp*. Terakhir, sebagai contoh, pemeriksa diperbolehkan untuk memastikan hubungan antara dua R_i dengan segera tanpa harus melakukan pengecekan *time stamp* satu per satu, namun sebuah *time stamp* (*individual time stamp*) dari *round r* tidak dihubungkan dengan R_{r-1} dari *round* sebelumnya, seperti terlihat dalam gambar 11.

5.3 *Binary Linking Scheme*

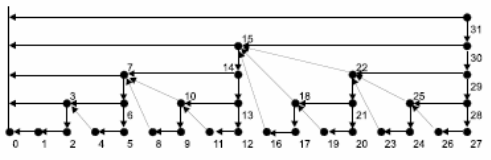
Binary linking scheme dapat didefinisikan sebagai sebuah graf berarah dan terhingga yang saling terhubung, tidak mengandung *cycle* (*loop*) dalam graf dan semua puncak (*vertex*) mempunyai dua anak panah (*link*) yang mengarah keluar. Untuk membuat sebuah graf tersebut, sebut saja T_k, dilakukan cara-cara sebagai berikut:

1. T_1 terdiri dari sebuah puncak (*vertex*) yang dinamakan dengan nomor 1. *Vertex* ini merupakan *source* (awal) dan *sink* (akhir) dari graf T_1.
2. Misalkan T_k telah dibentuk, *sink*-nya dinamakan $2^k - 1$. Graf T_{k+1} terdiri dari dua salinan graf T_k, dimana *sink* dari salinan kedua dihubungkan dengan *source* dari salinan pertama, dan sebuah *vertex* tambahan $2^{k+1} - 1$ yang dihubungkan dengan *source* dari salinan kedua. Nama dari salinan kedua ditambah dengan $2^k - 1$. *Sink* dari T_{k+1} sama dengan *sink* dari salinan pertama, *source* dari T_{k+1} sama dengan *vertex* yang dinamakan $2^{k+1} - 1$. Setelah itu, hubungkan semua *vertex-vertex* dari salinan kedua yang mempunyai jumlah *link* yang menuju keluar kurang dari dua, ke *source* dari salinan pertama. Ingat bahwa sekarang terdapat sebuah *double link* dari *sink* salinan kedua ke *source* salinan pertama.



Gambar 12: Gambaran dari pembentukan sebuah *binary linking scheme*

Rangkaian (T_k) mendefinisikan sebuah *binary linking scheme* yang dalam setiap *scheme*-nya mengandung *scheme* T_k sebagai bagian awal dan terdpat *vertex-vertex* yang dinamakan dengan nomor. Setelah pembentukan dari *binary linking scheme* ini, harus ditambahkan *link-link* dari *source*-nya bagian awal manapun ke sebuah *vertex* spesial yang dinamakan dengan 0, lihat gambar 13.



Gambar 13: *Binary linking scheme*

5.4 *Synchronized Tree Scheme*

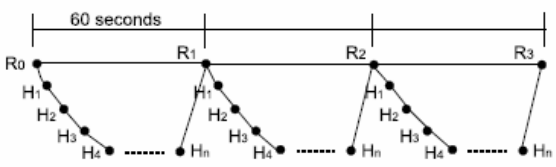
Selanjutnya, sebuah evolusi dari metode yang telah dipuji dan dihargai akan dijelaskan dalam

scheme ini. Ini telah memperbaiki dan meningkatkan performansi dari proses *Digital Time-Stamping*. Dengan mengambil kelebihan dari konsep-konsep sebelumnya, sebuah *linking scheme* yang efisien dapat dibuat untuk mengurangi waktu mendapatkan sebuah *time stamp* individu.

Dengan mempertimbangkan semua nilai *hash* yang tiba untuk diberi *time stamp* dan saling terhubung, serta pada akhir waktu, sebuah *function F* diterapkan pada semua nilai *hash* tersebut untuk menghasilkan hanya sebuah *time stamp* yang merepresentasikan semua *time stamp* tersebut, maka dapat dibuat *scheme* sebagai berikut:

1. *Customer* yang ingin memberi *time stamp* dokumennya, menghasilkan nilai *hash* dan mengirimkannya ke *Time-Stamping Service*.
2. Ketika nilai *hash* akan diberi *time stamp*, nilai tersebut dihubungkan dengan daftar permintaa *time stamp* lainnya melalui sebuah *function F*.
3. Pada akhir setiap *round*, sebagai contoh 60 detik, *function F* diterapkan kembali dengan elemen terkahir dari rantai dan dengan *time stamp* dari *round* sebelumnya, ini menghasilkan *time stamp* dari *round* tersebut, lihat gambar 14.

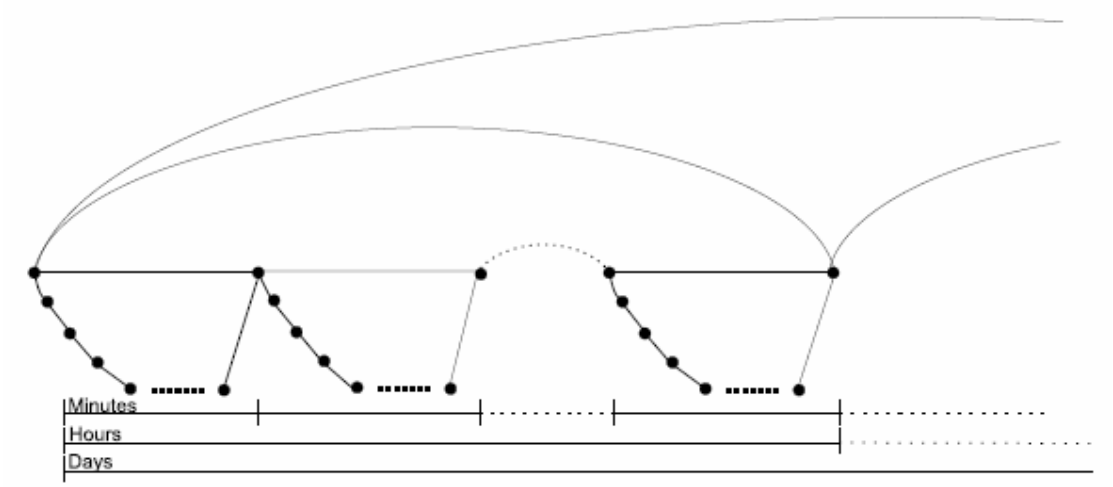
Setiap H_i adalah representasi dari setiap nilai *hash* yang diterima *Time-Stamping Service* dan dihubungkan dengan sebuah *function F*. R_0 adalah *time stamp* dari *round* sebelumnya dan R_1 adalah *time stamp* pada saat itu dari *round* dan seterusnya.



Gambar 14: Contoh dari sebuah *round* sederhana.

Untuk membuat proses menjadi lebih umum (perluasan), *round* yang dibuat harus lebih besar, representasi dari unit-unit waktu yang diketahui seperti jam, hari, bulan, dan tahun, untuk mengurangi waktu pemeriksaan dari dokumen. Perluasan ini dapat dilihat pada gambar 15.

Pemeriksaan dapat diselesaikan dengan mencari *tree* bentukan, berdasarkan kepada waktu dan nilai dari *time stamp* yang dihasilkan. Sebagai contoh, sebuah dokumen yang diberi *time stamp* 22/03/2001 dan 14:32 PM, urutan dari pemeriksaan adalah tahun, bulan, tanggal, jam, menit, dan kemudian nilai *hash* dari dokumen diperiksa. Perlu diingat bahwa semua metode atau *scheme* yang disebutkan diatas sama baiknya dengan metode ini, mempublikasikan semua *time stamp* pada sebuah direktori umum.



Gambar 15: Perluasan dari *scheme*

6. Penggunaan *Time Stamp*

Time stamp telah banyak digunakan dalam dokumentasi suatu data yang memerlukan keabsahan dari waktu untuk menunjukkan bahwa data tersebut adalah sah. Krabsahan ini menjamin bahwa isi dari data tidak pernah diubah sejak diberi *time stamp* dan waktu yang

terdapat pada *time stamp* menunjukkan waktu sesungguhnya.

Banyak data yang menggunakan *time stamp* dalam menjaga keamaannya. Salah satunya adalah dokumen mengenai suatu penelitan yang memerlukan pengakuan atas waktu dari penelitiannya. Hal ini untuk menjaga agar tidak ada peneliti lain yang mengaku melakukan penelitian yang sama dan waktunya lebih dulu dari dokumen penelitian yang sebenarnya. Salah satu aplikasi yang dapat digunakan dalam pemberian *time stamp* adalah Adobe Acrobat versi 7 dimana *time stamp server* harus dikonfigurasi terlebih dahulu sebelum menggunakannya.

Pada bidang lain, misalnya lembaga keuangan, laporan keuangan yang dikeluarkan setiap tahunnya harus dijamin bahwa waktu yang tertera pada laporan tersebut adalah yang sebenarnya. Selain itu, dapat diterapkan pula pada transaksi keuangan yang terdapat dalam lembaga keuangan dengan tujuan waktu dari transaksi menyatakan waktu sesungguhnya dan menjamin tidak terjadi manipulasi.

Penggunaan *time stamp* yang lain adalah *IP telephony*. Sebagai contoh, *Real-time Transport Protocol* (*RTP*) memberikan rangkaian (*sequential*) *time stamp* pada paket-paket suara sehingga mereka dapat ditampung dalam sebuah *buffer* oleh penerima, dirakit ulang, dan terkirim tanpa adanya kesalahan karena waktu dari setiap paket suara yang dikirimkan tidak mungkin berubah selama pengiriman.

Dalam melakukan transaksi keuangan yang berbasis *e-commerce*, *time stamp* dapat pula digunakan dengan tujuan untuk menghindari adanya pesanan palsu atau tidak sah. Tujuan lain adalah untuk menghindari terjadinya perselisihan antara *customer* dengan perusahaan *e-commerce* itu sendiri.

Masih terdapat penggunaan *time stamp* lainnya, yang sangat berguna untuk menghindari kesalahpahaman antara dua atau beberapa pihak mengenai keabsahan suatu dokumen. Suatu dokumen digital diberi *time stamp* agar suatu dokumen diakui keabsahannya baik waktu maupun isi dari dokumennya.

7. Masalah yang muncul dalam *Time Stamp*

Semua tahapan implementasi dari *Digital Time-Stamping* perlu mendapat perhatian, dari pembuatan nilai *hash* dari dokumen sampai pada penerimaan *time stamp* oleh *customer*. Diantara tahapan-tahapan ini, terdapat banyak kemungkinan yang dapat mengakibatkan persoalan atau kesalahan. Beberapa kemungkinan tersebut dijelaskan sebagai berikut:

1. *Spoofing*: merupakan sebuah serangan yang terjadi ketika sebuah *host* memalsukan diri menjadi yang lain dengan tujuan untuk menangkap pesan yang diperuntukan untuk mesin yang asli. Pada *Digital Time-Stamping*, serangan ini dapat terjadi dimana sebuah mesin berpura-pura menjadi entitas *Time-Stamping* dan menghasilkan *time stamp* yang salah. Permasalahan ini dapat diselesaikan dengan memperkenalkan sebuah sistem yang dapat mendeteksi serangan tersebut, biasanya dinamakan *Intruders Detection System* (*IDS*).
2. *Denial of Service*: adalah sebuah kemampuan satu atau beberapa *host* untuk mengirimkan beberapa nilai *hash* ke sistem yang terbuka dari korban, dalam kasus ini, entitas *Time-Stamping*. Dengan cara ini, penyerang (*attacker*) memenuhi koneksi dari *Time-Stamping Service* untuk memaksanya menolak permintaan *time stamp* yang baru. Penyelesaian akan masalah ini sama dengan *Spoofing*.
3. *Cheat the stamp's chain*: seseorang yang mempunyai akses terhadap entitas *Time-Stamping* dan mengetahui bagaimana *linking* dari *time stamp*, dapat membuat sebuah cabang rantai yang salah dari rantai tersebut untuk mendapatkan hasil yang salah. Hal ini dapat diselesaikan dengan menjalankan pencarian secara periodik di dalam rantai atau *tree* untuk memeriksa jika terdapat cabang yang salah.
4. *Compromised function F*: ketika hal ini terjadi, entitas *Time-Stamping* sebaiknya melakukan prosedur perbaikan, jika tidak dilakukan maka tidak terdapat kepercayaan *time stamp* yang dibuat oleh entitas tersebut. Tapi entitas *Time-Stamping* tidak membutuhkan semua dokumen untuk melakukan proses *linking* lagi. Salah satu solusinya adalah memakai dua tipe *function F*, sebagai contoh, nilai *hash* dihasilkan dengan MD5 dan juga dengan SHA-1, sehingga membuatnya semakin susah bagi penyerang untuk memalsukan sebuah

time stamp. Solusi yang lain adalah secara periodik menandatangani kembali semua *node-node* dari rantai dengan metode tanda tangan yang lebih baru.

8. Kesimpulan

Penggunaan dokumen digital yang sah pada saat ini telah menjadi suatu hal yang tidak bisa dihindari. Setiap orang menginginkan dokumen yang diterimanya mengandung informasi yang sesungguhnya dan *valid*.

Time stamp sebagai salah satu metode dalam kriptografi dapat digunakan untuk mengatasi masalah tersebut. Pemberian tanda tangan dan waktu dapat digunakan sebagai suatu tanda yang membuktikan bahwa suatu dokumen adalah sah.

Pemberian *time stamp* dapat dilakukan oleh seorang pihak yang dipercaya, biasanya disebut *Time-Stamping Authority*, atau oleh beberapa orang dalam sebuah kelompok (*distributed trust*). Masing-masing tipe ini mempunyai kelebihan dan kekurangan. Akan tetapi, pada saat ini, penggunaan *Time-Stamping Authority* lebih umum dan dapat dipercaya.

Layanan yang diberikan oleh *Time-Stamping Authority* disebut dengan *Time-Stamping Service*. Layanan ini yang memberi *time stamp* pada setiap nilai *hash* yang dikirim oleh *customer* dan menjaga keabsahan dari setiap *time stamp* yang dihasilkan.

Terdapat beberapa *scheme* yang dapat digunakan oleh *Time-Stamping Authority* dalam menjaga keabsahan dari *time stamp* yang telah diberikan. *Scheme* yang dapat digunakan adalah *linking scheme*, *tree scheme*, *binary linking scheme*, dan *synchronized scheme*.

Time stamp telah banyak digunakan dalam berbagai bidang. Biasanya digunakan untuk menjamin kebenaran waktu pembuatan sebuah dokumen dan menjaga kebenaran dari isi dokumen tersebut. Salah satu contoh aplikasinya adalah bidang penelitian yang banyak menghasilkan dokumen mengenai suatu penemun dimana waktu pembuatan dokumen sangat penting dan perlu dijaga sehingga tidak ada pihak manapun yang dapat mengubah isinya atau mengakui dokumen tersebut merupakan miliknya padahal bukan.

Penerapan *time stamp* memunculkan suatu permasalahan yaitu bagaimana mempunyai sebuah sistem yang dapat diandalkan dengan memperhatikan faktor lainnya. Permasalahan ini tidak hanya tergantung pada *scheme* apa yang digunakan tapi terdapat aspek lain yang berperan dalam pemecahan permasalahan ini.

Selain itu, terdapat beberapa pertimbangan lain mengenai masalah keamanan dari *scheme-scheme* yang digunakan. Namun, semua permasalahan yang mungkin timbul dapat diselesaikan jika *Time-Stamping Authority* menangani masalah keamanan dengan serius dan melakukan pengecekan secara periodik pada *scheme*.

DAFTAR PUSTAKA

[1] Haber, Stuart and Scott Stornetta, W. *How to Time-Stamp a Digital Document*. http://www.cs.utk.edu/~dunigan/cns04/timestamp.pdf. Tanggal akses: 8 November 2006 pukul 15.00.

[2] Bayer, Dave and Haber, Stuart. *Improving the Efficiency and Reliability of Digital Time-Stamping*. http://www.math.columbia.edu/~bayer/papers/Timestamp_BHS93.pdf. Tanggal akses: 8 November 2006 pukul 15.00.

[3] Munir, Rinaldi. (2006). *Bahan Kuliah IF5054 Kriptografi*. Departemen Teknik Informatika, Institut Teknologi Bandung.

[4] What is timestamp? – a definition from Whatis.com – see also: time stamp. http://whatis.techtarget.com/definition/0,,sid9_gci817089,00.html. Tanggal akses: 8 November 2006 pukul 15.00.

[5] Pasqual, Everton Schonardie, Dias, Julio, and Custodio, Ricardo Felipe. *A new method for Digital Time-Stamping of Electronic Document*. http://www.bry.com.br/downloads/artigo/Digital%20Time-Stamping%20of%20Electronic%20Document.pdf. Tanggal akses: 8 November 2006 pukul 15.00.

[6] Wouters, Karel. Time Stamping: a survey. http://www.esat.kuleuven.ac.be/cosic/seminars/slides/Time-Stamping.pdf. Tanggal akses: 8 November 2006 pukul 15.00.

[7] Protocols and data formats for time-stamping service. http://ltans.edelweb.fr/Cybernetica-OpenEvidence-TimeStamping.pdf.

Tanggal akses: 8 November 2006 pukul 15.00.

[8] Weibel, Hans and Dominic Béchaz. Implementation and Performance of Time Stamping Techniques. Zurich University of Applied Sciences, Winterthur, Switzerland. Tanggal akses: 8 November 2006 pukul 15.00.

[9] Timestamp Definition – encyclopedia of pcmag.com. *http://www.pcmag.com/encyclopedia_term/0,2542,t=time+stamp&i=52924,00.asp*. Tanggal akses: 8 November 2006 pukul 15.00.

[10] Timestamp definition by The Linux Information Project. *http://www.bellevuelinux.org/timestamp.html*. Tanggal akses: 8 November 2006 pukul 15.00.

[11] How a digital time stamp works. *http://www.digistamp.com/timestamp.htm*. Tanggal akses: 8 November 2006 pukul 15.00.

[12] Adobe Acrobat Time Stamp Servers. *http://www.digistamp.com/acrobat.htm*. Tanggal akses: 8 November 2006 pukul 15.00.

[13] Time stamp – Certum Time-Stamping Authority Non-Repudiation System. *http://www.certum.pl/english/eng/services/ts/zasady.html*. Tanggal akses: 8 November 2006 pukul 15.00.

[14] Cryptography in Everyday Life. *http://www.eco.utexas.edu/faculty/Norman/BUS.FOR/course.mat/SSim/life.html*. Tanggal akses: 8 November 2006 pukul 15.00.

[15] Wikipedia - Timestamp. *http://en.wikipedia.org/wiki/Digital_timestamping*. Tanggal akses: 8 November 2006 pukul 15.00.

[16] Wikipedia - Digital Signature. *http://en.wikipedia.org/wiki/Digital_signature*. Tanggal akses: 8 November 2006 pukul 15.00.