ISSN 2085-4552

Implementasi Keamanan pada Transaksi Data Menggunakan Sertifikat Digital X.509

Is Mardianto[1], Kuswandi[2]

[1,2] Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Trisakti, Jakarta, Indonesia
mardianto@trisakti.ac.id[1]

Diterima 3 Pebruari 2016
Disetujui 1 Juni 2016

***Abstract*—Security issues have become a major issue on the Internet. One of the security methods that are widely used today is to implement a digital certificate. Digital certificates have evolved over time, one of which is the X.509 digital certificate. Digital certificates have been widely used as authentication applications, web network authentication and other authentication systems that require digital certificates.**

This research is carried out by implementing an X.509 digital certificate technology as a *mobile web service* with its client. Secure Hash Algorithm (SHA), Diffie-Hellman, and Advanced Encryption Standard (AES) are used to secure the data exchange transaction between the *web service* and *mobile* phone. SHA algorithm will be used for user authentication, Diffie-Hellman algorithm will be used for public key exchange and AES algorithms will be used for symmetric cryptography data.

The results of the application of digital certificates, the SHA algorithm, Diffie-Hellman, and AES in *mobile* phone applications, provide security application running on *web service*.

***Index Terms*—Digital Certificate, X.509, SHA, Diffie Hellman, AES**

I. Pendahuluan

Masalah keamanan data dalam berkomunikasi saat ini menjadi hal yang sangat penting, terutama dalam hal transaksi data di antara pengguna telepon pintar. Salah satu penerapan *business logic* yang menggunakan jaringan internet adalah transaksi jual-beli barang, transaksi keuangan, pemesanan tiket, dan lainnya. Selain itu ada pula jenis-jenis layanan dalan jaringan internet, salah satunya adalah *web service*.

Web service merupakan suatu bagian dari *business logic* yang terletak di suatu tempat di internet yang dapat diakses melalui standar internet protokol seperti HTTP atau SMTP. Penggunaan *web service* dapat dilakukan secara sederhana, seperti mengunjungi suatu *website* atau secara kompleks seperti fasilitas negosiasi bisnis multi organisasi. Agar efisiensi dan fleksibilitas dapat tercapai, banyak pengembang telah meneliti dan menciptakan teknologi yang dapat memenuhi dua syarat di atas. Teknologi tersebut adalah perangkat telepon pintar.

Latar belakang implementasi pada perangkat aplikasi *mobile* karena semakin banyak pengguna yang lebih memilih kemudahan dan kesederhanaan dalam mengakses aplikasi. Dengan adanya perangkat telepon pintar, pertukaran informasi dapat berkembang lebih pesat dibandingkan dengan teknologi sebelumnya, ini dikarenakan sifat mobilitas yang dimilikinya. Karena selain fungsi utamanya untuk berkomunikasi, alat ini memiliki beberapa fungsi lain yang dapat memberikan hiburan dan pengaksesan informasi melalui akses internet. Dengan adanya fitur ini, telepon pintar dapat mengakses informasi dengan lebih luas dan fleksibel. Dengan kebutuhan informasi yang tinggi secara *mobile* inilah, persoalan keamanan menjadi hal yang penting, Karena sifat publisitas sistem *mobile enterprise* membuat semakin perlunya keamanan data.

Untuk masalah keamanan dalam pengaksesan suatu *web service*, salah satu metode yang dapat digunakan adalah dengan cara sertifikat digital. Metode ini sudah banyak digunakan dalam perusahaan-perusahaan besar dengan tujuan agar pengunjung *web service* yakin bahwa web yang dituju adalah web yang benar sehingga pengguna dapat melakukan transaksi tanpa ragu. Selain sertifikat digital, dibutuhkan juga algoritma-algoritma kriptografi untuk mengamankan data yang akan dikirim dan diterima oleh pengguna dan *web service* itu sendiri. Dengan kebutuhan keamanan inilah telah dimunculkan banyak algoritma untuk mengenkripsi data, di antaranya algoritma kriptografi simetrik AES.

Berdasarkan latar belakang tersebut, maka permasalahan yang muncul untuk dijawab adalah bagaimana cara mengimplementasikan keamanan pada sistem *web service* berbasis perangkat *mobile* dengan sistem otentikasi sertifikat digital dan menggunakan algoritma kriptografi simetrik sebagai metode keamanan pertukaran data.

Sertifikat digital digunakan agar pengguna yakin bahwa *web server* yang dituju adalah *web server* yang benar demikian juga sebaliknya dari sisi *web server*, yakin bahwa pengguna yang telah mengakses merupakan pengguna dari *web service* yang telah terdaftar.

II. Tinjauan Pustaka

A. Kriptografi Kunci Publik

Kriptografi kunci publik adalah algoritma yang menggunakan dua kunci yang berbeda untuk melakukan proses kriptografi[2], kebalikan dari metode enkripsi simetrik, yang menggunakan hanya satu kunci saja. Penggunaan dua kunci tersebut memiliki konsekuensi pada sifat kerahasiaan dan distribusi kunci.

B. Pertukaran Kunci Diffie-Hellman

Proses pertukaran kunci diilustrasikan pada gambar 1 dan 2[1]. Pengguna A memilih bilangan bulat acak $X_A < q$ mengkomputasikannya $Y_A = \alpha^{X_A} mod\ q$. Begitu juga dengan B, memilih bilangan bulat acak $X_B < q$ dan mengkomputasikannya $Y_B = \alpha^{X_B} mod\ q$ Setiap sisi menyimpan nilai privat X dan membuat nilai Y tersedia secara publik. A lalu menghitung kunci

$$K = (Y_B)^{X_A} mod\ q \quad (1)$$

dan B melakukan komputasi kunci

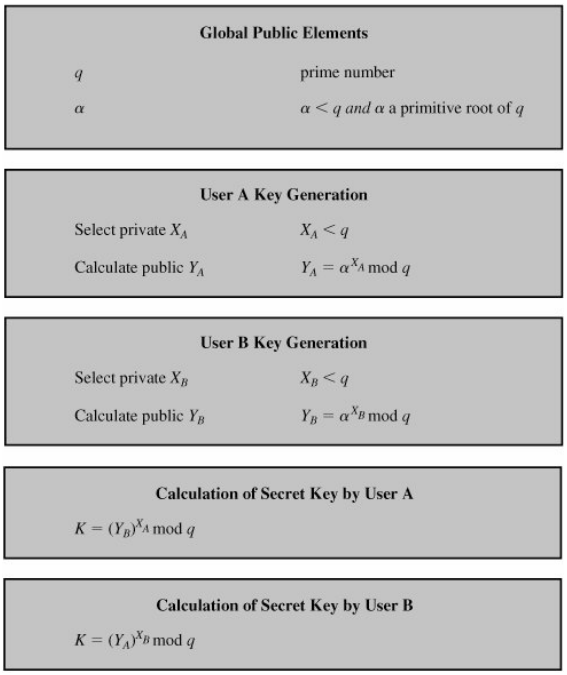
$$K = (Y_A)^{X_B} mod\ q. \quad (2)$$

Dua perhitungan tersebut akan menghasilkan nilai kunci K yang sama

$$\begin{aligned} K &= (Y_B)^{X_A} mod\ q \\ &= (\alpha^{X_B} mod\ q)^{X_A} mod\ q \\ &= (\alpha^{X_B})^{X_A} mod\ q \\ &= (\alpha^{X_A})^{X_B} mod\ q \\ &= (\alpha^{X_A} mod\ q)^{X_B} mod\ q \\ &= (Y_A)^{X_B} mod\ q. \end{aligned}$$

Hasil dari komputasi kedua sisi menghasilkan pertukaran nilai rahasia. Karena nilai X_A dan X_B adalah nilai yang bersifat privat, walaupun musuh/ *adversary* mengetahui nilai q, α, Y_A, Y_B. Maka musuh tidak dapat mengetahui nilai kunci K yang digunakan user A dan B terkecuali melakukan komputasi *discrete logarithm* yang mendekati sifat mustahil untuk dikerjakan.

$$X_B = dlog_{\alpha,q}(Y_B) \quad (3)$$



Gambar 1. Algoritma Pertukaran Kunci Diffie Hellman [1]

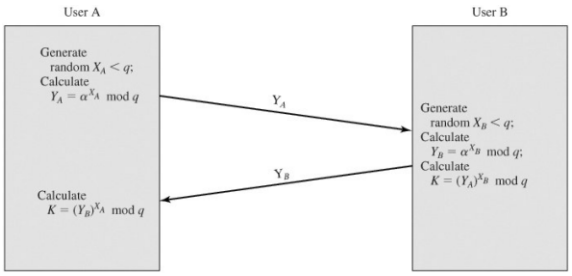
C. Sertifikat Digital X.509

Inti dari skema X.509 adalah sertifikat kunci publik yang diasosiasikan pada setiap pengguna. Pengguna dari sertifikat ini diasumsikan membuat sertifikat dari suatu otoritas penerbit sertifikat (*Certificate Authority*/ CA). Direktori *server* menyediakan tempat yang mudah diakses oleh pengguna untuk mendapatkan sertifikat. Format umum dari sertifikat seperti terlihat pada Gambar 3), di antaranya: [1]

- *Version*: Pada umumnya sertifikat adalah versi 1. Jika *Issuer Unique Identifier* atau *Subject Unique Identifier* terdapat dalam sertifikat, maka versi sertifikat tersebut adalah versi 2. Jika ada satu atau lebih tambahan, maka versi sertifikat tersebut adalah versi 3.
- *Serial number*: sebuah nilai bilangan bulat, yang unik yang dikeluarkan oleh CA.
- *Signature algorithm identifier*: algoritma yang digunakan untuk menandatangani sertifikat, beserta parameter yang terkandung dalam sertifikat.
- *Issuer name*: nama X.500 dari CA yang membuat dan menandatangani sertifikat.
- *Period of validity*: Terdiri dari dua tanggal, masa pembuatan dan kadaluarsa dari sertifikat.
- *Subject name*: nama dari pengguna yang membuat sertifikat. Sertifikat ini menjamin kunci publik dari pemegang kunci privat.
- *Subject's public key information*: kunci publik dari subjek ditambah dengan identifikasi dari algoritma yang digunakan.
- *Issuer unique identifier*: pilihan tambahan yang

digunakan sebagai identifikasi secara unik mengenai CA.

- *Subject unique identifier*: pilihan tambahan yang digunakan sebagai identifikasi secara unik mengenai pemegang sertifikat.
- *Extensions*: kumpulan dari satu atau lebih bagian tambahan. Tambahan ini terdapat pada sertifikat versi 3.
- *Signature*: meliputi seluruh bagian dari sertifikat; yang berisi kode hash yang dienkripsi oleh kunci privat milik CA. Pada bagian ini juga terdapat identifikasi algoritma signature.



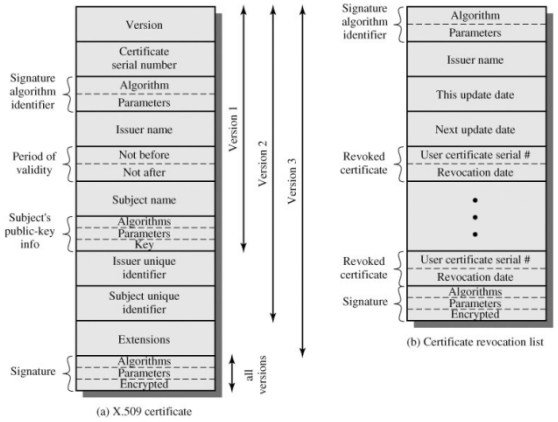
Gambar 2. Prosedur pertukaran kunci Diffie-Hellman [1]

D. Advanced Encryption Standard (AES Cipher)

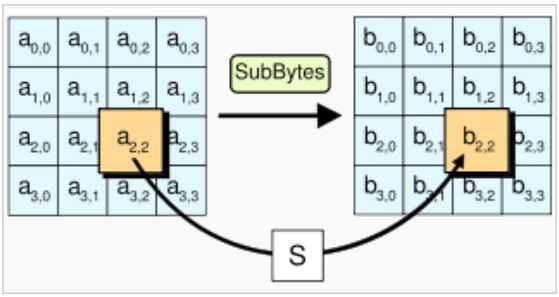
Algoritma AES pertama kali dipublikasikan oleh National Institute of Standards and Technology (NIST) di AS, 26 November 2001[5].

Karakteristik dan prosedur algoritma AES di antaranya [1] :

1. Satu fitur yang penting dari struktur ini adalah bahwa struktur ini bukan struktur Feistel. Jika dilihat dari struktur Feistel yang lama, setengah dari blok data digunakan untuk memodifikasi setengah dari blok data selanjutnya, lalu setengahnya lagi ditukar. Dua bagian dari AES tidak menggunakan struktur Feistel tetapi keseluruhan proses blok data dilakukan secara paralel dengan menggunakan subtitusi dan permutasi.
2. Kunci yang digunakan sebagai kunci masukan diubah menjadi *array* dari empat puluh empat bilangan 32 bit, w[i]. Setiap elemen *array*, yang terdiri dari 4 elemen, melakukan fungsi kunci *round* untuk setiap *round*.
3. Empat proses dari tiap bagian proses *round*:
   - Subtitusi *byte* (Gambar 4): menggunakan S-box untuk melakukan subtitusi dari *byte* ke *byte* dari blok satu ke blok lainnya.

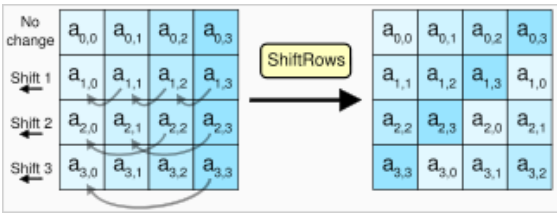


Gambar 3. Format X.509[1]



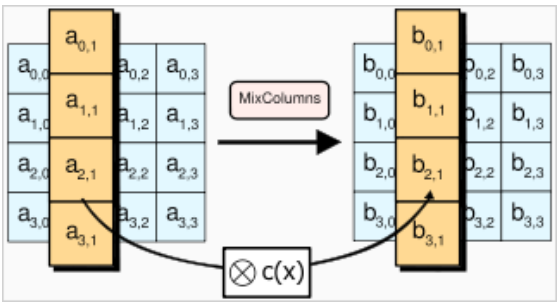
Gambar 4. Proses Substitution *Bytes*[1]

- ShiftRows: sistem permutasi yang sederhana (Gambar 5).



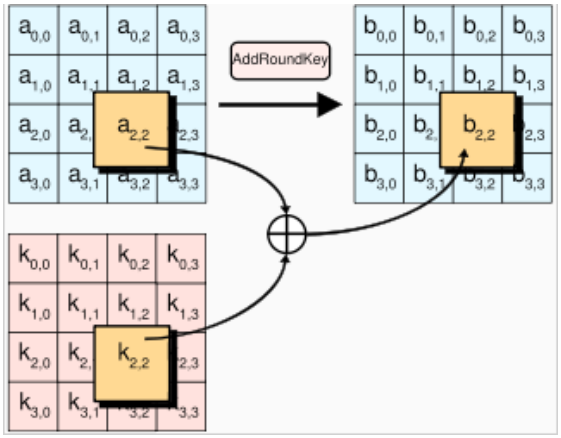
Gambar 5. Proses Shift Rows[1]

- MixColumns: subtitusi yang menggunakan aritmatik melalui GF(), pada Gambar 6.



Gambar 6. Proses Mix Columns[1]

- *AddRoundKey* : proses XOR sederhana pada setiap blok (Gambar 7).



Gambar 7. Proses Add *Round* Key[1]

4. Antara enkripsi dan dekripsi selalu dimulai dengan bagian *AddRoundKey*, yang diikuti dengan sembilan *round* yang masing-masing terdiri dari empat bagian proses dan sepuluh *round* yang masing-masing terdiri dari tiga bagian proses.
5. Hanya bagian *AddRoundKey* yang menggunakan kunci (Gambar 7). Untuk alasan ini, cipher yang dimulai dan diakhiri dengan proses pada bagian *AddRoundKey*. Untuk bagian lain digunakan saat awal atau akhir saja.
6. Fungsi kebalikan digunakan untuk proses dekripsi. Untuk bagian *AddRoundKey*, kebalikan dapat dicapai dengan proses XOR sekali lagi dengan kunci *round* yang sama pada blok.

E. Secure Hash Algorithm (SHA)

SHA merupakan salah satu fungsi hash satu arah yang telah dikembangkan oleh NIST dan dipublikasikan oleh Federal Institute Processing Standard (FIPS 180) pada tahun 1993. Versi revisi dipublikasikan sebagai FIPS 180-1 pada tahun 1995 dan secara umum dikenal sebagai SHA-1[4].

Berikut tahap-tahap yang menggambarkan kerja dari SHA (Gambar 8): [4]

1. Penambahan bit penjanggal

Pesan ditambahkan sejumlah bit penjanggal sehingga panjang pesan kongruen dengan 896 modular 1024. Penambahan ini selalu dilakukan meskipun panjang pesan sudah memenuhi. Jadi bilangan bit yang ditambahkan berkisar 1 hingga 1024.

2. Penambahan panjang pesan

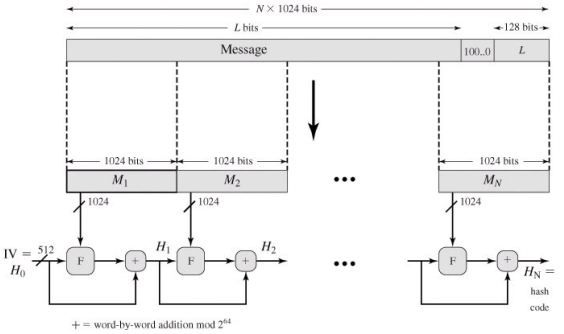
Blok 128 bit ditambahkan ke dalam pesan. Blok ini berisi 128 bit yang mewakili nilai bilangan bulat panjang dari pesan asli.

3. Inisialisasi nilai penyangga

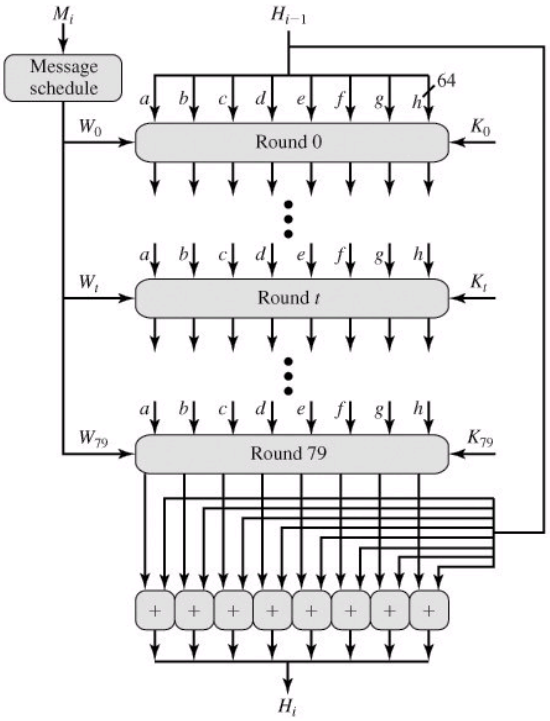
512 bit *buffer* digunakan untuk menangani hasil bagian pertengahan dan bagian akhir dari fungsi hash. *Buffer* ini dapat direpresentasikan sebagai delapan buah 64 bit (a, b, c, d, e, f, g, h). Bagian ini diinisialisasi dengan bilangan bulat 64 bit.

4. Pengolahan pesan dalam blok 1024 bit

Gambar 9 dapat menjelaskan bagian pengolahan pesan, yang pada intinya pengolahan ini terdiri dari 80 proses *round*s. Pada Gambar 8, proses pengolahan ini diberi label F.



Gambar 8. Proses penyiapan blok SHA[1]



Gambar 9. Algoritma SHA[1]

F. Web service

Web service memiliki tiga komponen utama: [3]

1. *Universal Description, Discovery, and Integration* (UDDI)

UDDI yang merupakan suatu container pada *web service*. UDDI menyediakan mekanisme pencarian *web service*. Sebagian *server* J2EE sudah mengimplementasikan UDDI ini.

2. *Web service Description Language* (WSDL)

Sebelum pengguna melakukan permintaan ke

web service, pengguna harus mengetahui skema dari *web service*. Skema *web service* harus menunjukkan atribut-atribut *web service*. Atribut-atribut tersebut antara lain: fungsi-fungsi yang dimiliki lengkap dengan tipe data dan nilai pengembaliannya, parameter-parameter dari fungsi-fungsinya lengkap dengan skema tipe data. Standar skema *web service* adalah WSDL yang terbentuk dari XML. WSDL menyediakan pengetahuan kepada pengguna atas parameter, nilai pengembalian, dan tipe data dari atribut *web service*.

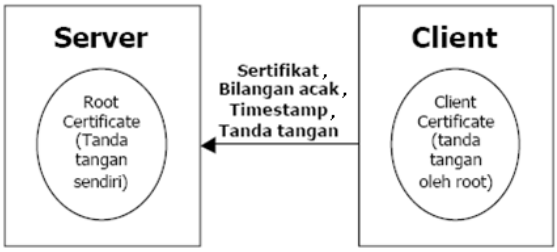
3. *Simple Object Access Protocol* (SOAP)

SOAP merupakan standar protokol objek pada *web service*, yang juga berbentuk XML, yang bertugas untuk mengantarkan objek *web service* dari *server* ke pengguna. SOAP tidak terikat oleh protokol transport apapun. Asalkan suatu protokol bisa mengirim teks, maka protokol tersebut bisa dipakai sebagai alat transport SOAP. Protokol yang dapat dipakai SOAP antara lain: HTTP, SMTP, dan sebagainya. SOAP terdiri dari header dan body yang terbungkus dalam isi pesan (*envelope*).

III. Analisa dan Perancangan

A. Mekanisme Otentikasi Pengguna

Mekanisme otentikasi pengguna yang dilakukan adalah dengan 2 metode, yaitu dengan sistem otentikasi *password* pengguna dan juga sistem otentikasi sertifikat digital. Gambar 10 merupakan gambaran mengenai sistem otentikasi menggunakan sertifikat digital.

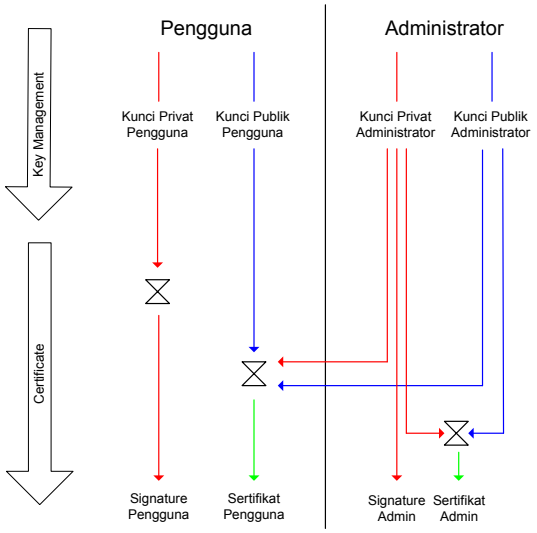


Gambar 10. Proses Otentikasi Sertifikat

B. Mekanisme Pembuatan Sertifikat X.509

Pembuatan sertifikat X.509 (Gambar 11) dilakukan oleh pengguna dengan cara membuat *Certificate Signing Request* (CSR), dimana CSR ini akan dikirim ke CA. Lalu CA akan menandatangani permintaan pengguna sehingga akan menghasilkan sertifikat yang telah ditandatangani untuk pengguna.

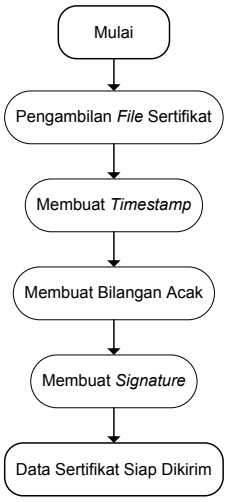
Teknik yang digunakan dalam penelitian ini pengguna tidak perlu membuat CSR karena CA itu sendiri adalah *administrator* dari aplikasi. Jadi prosedur yang dilakukan adalah pengguna mendatangi *administrator* dengan menyertakan data dirinya. Data ini akan dimasukkan ke dalam *database web service*, sehingga sinkronisasi data pengguna dan *web service* dilakukan pada proses pendaftaran ini.



Gambar 11. Proses Pembuatan Sertifikat Digital X.509

C. Alur Sertifikat Pada Pengguna

Sertifikat pada pengguna disimpan dalam aplikasi yang akan dikirim ke *web service* untuk diverifikasi. Pada saat pengiriman, data yang dikirimkan ke *web service* tidaklah hanya sertifikat saja, tetapi ada pula data lain yang dikirim. Data selain sertifikat yang dimaksud adalah *timestamp* (penanda waktu), *random number* (bilangan acak), dan *signature* (tanda tangan digital). Ketiga data ini lah yang digunakan sebagai tambahan verifikasi pengguna pada *web service* (Gambar 12).

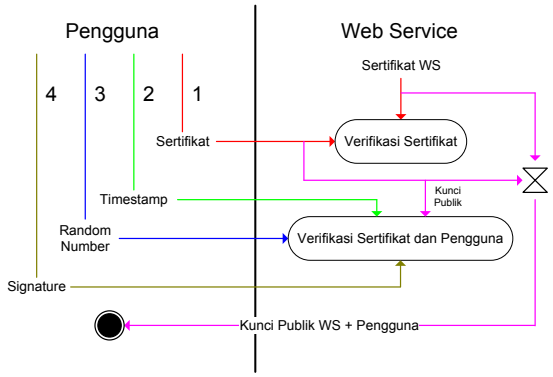


Gambar 12. Alur Data Sertifikat Pada Pengguna

D. Alur Sertifikat pada WebService

Pada sisi *web service*, sertifikat dan data lainnya yang diterima dari pengguna dalam bentuk bilangan heksadesimal akan diubah menjadi tipe data semula yaitu tipe data *byte* dalam array. Setelah itu sertifikat pengguna akan diubah menjadi format sertifikat X.509 untuk diverifikasi dengan sertifikat *root* milik *web*

service (Gambar 13). Untuk data lainnya tetap dalam bentuk *byte* dalam array lalu diproses sedemikian sehingga *web service* dapat memverifikasi *signature* dari sertifikat yang dikirim.



Gambar 13. Alur Sertifikat Pada *Web service*

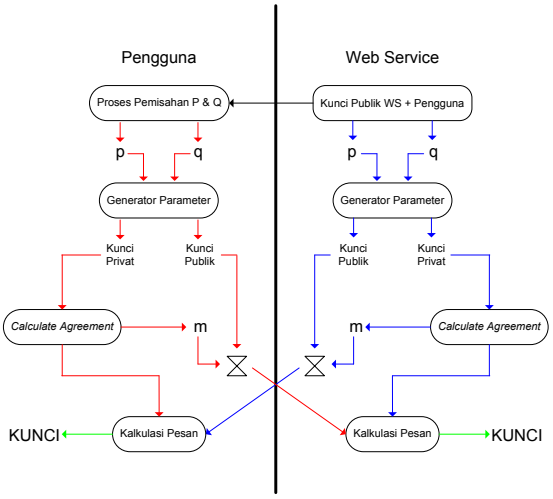
E. Proses Alur Pertukaran Kunci

Gambar 14 menjelaskan proses alur pertukaran kunci. Kunci publik *web service* dan pengguna yang berasal dari *web service*, yang merupakan kunci publik yang telah dipilah dari kunci publik milik masing-masing pihak. Setelah itu digabung dan dikirim ke pengguna.

Pengguna akan memisahkan data yang dikirim oleh *web service* menjadi dua konstanta yaitu: p dan q. Dua konstanta ini akan digunakan untuk menghasilkan parameter kunci privat dan kunci publik untuk melakukan pertukaran kunci. Lalu kunci privat digunakan untuk proses persetujuan dan kalkulasi pesan (m). Dua parameter yang dikirim ke *web service* untuk diproses adalah kunci publik dan kalkulasi pesan.

Setelah *web service* menerima permintaan dari pengguna, maka *web service* akan melakukan tindakan yang sama yaitu mengambil konstanta p dan q. Lalu melakukan proses pembentukan dua parameter kunci dan kalkulasi pesan. Setelah itu sebelum kunci publik dan kalkulasi pesan dikirim ke pengguna, *web service* akan memproses data yang dikirim oleh pengguna. Proses data ini melibatkan suatu pesetujuan dari kunci privat dari sisi *web service*, maka hasil proses ini akan menghasilkan sebuah kunci. Setelah proses perhitungan kunci selesai, kunci publik dan kalkulasi pesan dikirim ke pengguna.

Hal yang sama dilakukan oleh pengguna dalam memproses perhitungan kunci, yaitu data yang dikirim oleh *web service* akan diproses dan melibatkan suatu persetujuan dari pengguna. Maka hasil perhitungan tersebut akan menghasilkan kunci yang sama dengan hasil perhitungan yang dilakukan oleh *web service*.

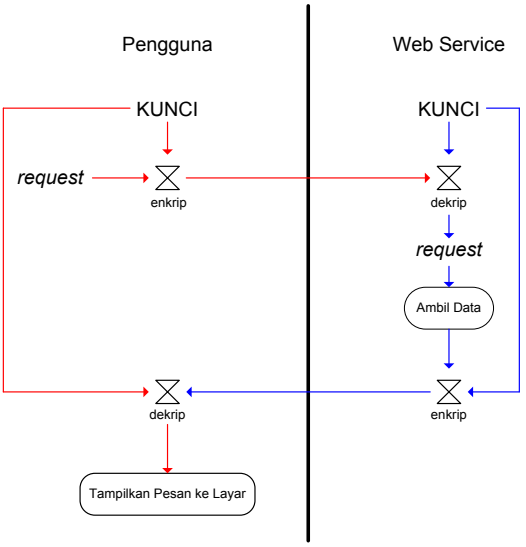


Gambar 14. Proses Alur Pertukaran Kunci

F. Proses Alur Pertukaran Data

Kunci yang terdapat pada pengguna dan *web service* berasal dari proses sebelumnya yaitu proses pertukaran kunci. Kunci ini digunakan untuk proses enkrip dekrip data antar pengguna dan *web service* (Gambar 15).

Pengguna akan mengenkrip pesan *request* dengan kunci, lalu dikirim ke *web service*. Pada *web service*, pesan ini akan didekrip dengan kunci yang ada. Setelah mengetahui apa permintaan dari pengguna, maka *web service* akan mengambil data yang diinginkan pengguna pada *database*. Data yang diambil ini akan dienkrip dengan kunci yang sama tadi, lalu dikirim ke pengguna. Maka pengguna perlu mendekrip pesan ini dengan kunci miliknya untuk menampilkan pesan yang terkandung.



Gambar 15. Proses Alur Pertukaran Data

IV. Uji Coba

A. Uji Coba Pembuatan Sertifikat

Admin akan memasukkan data yang diberikan oleh pengguna (Gambar 16). NIM dan *password* merupakan data yang dibutuhkan untuk memasuki sistem dan diverifikasi oleh *web service*.

Tombol "Create Cert Client" digunakan untuk membuat sertifikat dan kunci privat untuk pengguna dan tombol "Create WS Cert" digunakan untuk membuat sertifkat untuk *web service*.

Setelah selesai memasukkan data, lalu ditekan tombol "Create Cert Client" untuk membuat sertifikat. Maka akan muncul tampilan konfirmasi register pengguna.

Gambar 16. Tampilan Utama Register Member

Seperti penjelasan sebelumnya, bahwa sertifikat berisi informasi-informasi mengenai pengguna dari sertifikat. Selain itu juga berisi versi, nomer seri, algoritma tanda tangan digital, masa berlaku, penerbit, pemegang sertifikat, kunci publik dan informasi tambahan lainnya dari sertifikat yang terbentuk. Selain itu juga ada informasi mengenai jalur dari sertifikat tersebut, informasi ini biasanya digunakan untuk mengetahui bahwa siapa yang telah mensahkan sertifikat. Sehingga sertifikat akan jelas dari siapa dan untuk siapa sertifikat itu diberikan. Gambar 17, 18 dan 19 merupakan gambar-gambar bagian dari sertifikat yang berisi informasi dari sertifikat tersebut.

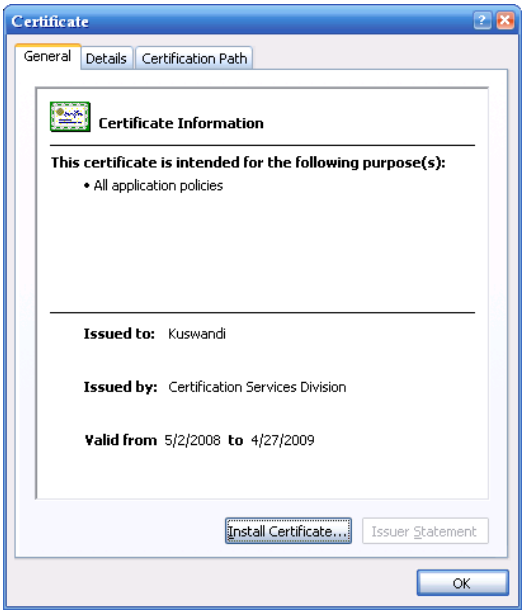
B. Uji Coba Sistem Web service

Uji coba ini berada pada tahapan *deployment*, hal ini merupakan langkah terakhir bagi para pengembang perangkat lunak. Tahap ini merupakan hal yang paling menentukan keberhasilan suatu sistem, karena sukses tidaknya aplikasi yang dikembangkan diukur melalui tahap implementasi dan uji coba ini. Implementasi yang yang dilakukan pada sistem *web service* ini adalah membuat *web service* dalam bentuk WSDL sehingga dapat diakses oleh pengguna.

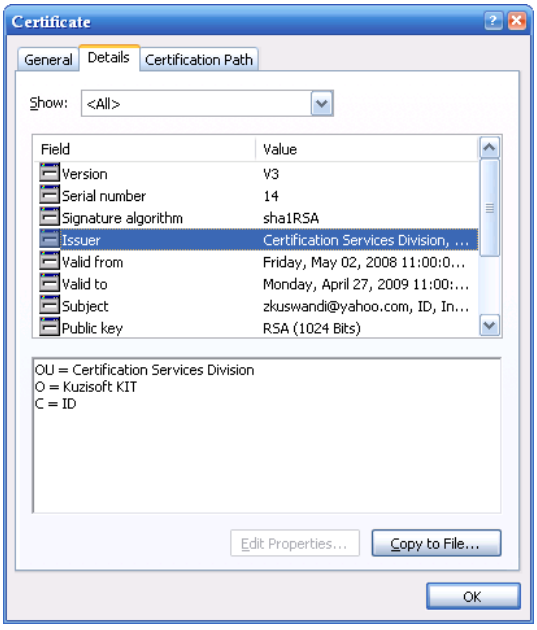
Berikut hasil dari WSDL yang telah terbentuk dan terdapat beberapa fungsi yang berfungsi untuk melayani permintaan penguna, di antaranya adalah:

1. cekLogin

Fungsi ini berfungsi untuk memverifikasi/ otentikasi pengguna yang ingin memasuki sistem. Data yang dikirim ke *web service* dari pengguna dimana data tersebut telah dienkrip dengan kriptografi SHA, lalu data tersebut akan diubah dari bilangan heksadesimal ke dalam bentuk *byte-byte*. Setelah itu data akan disesuaikan ke dalam *database web service*.



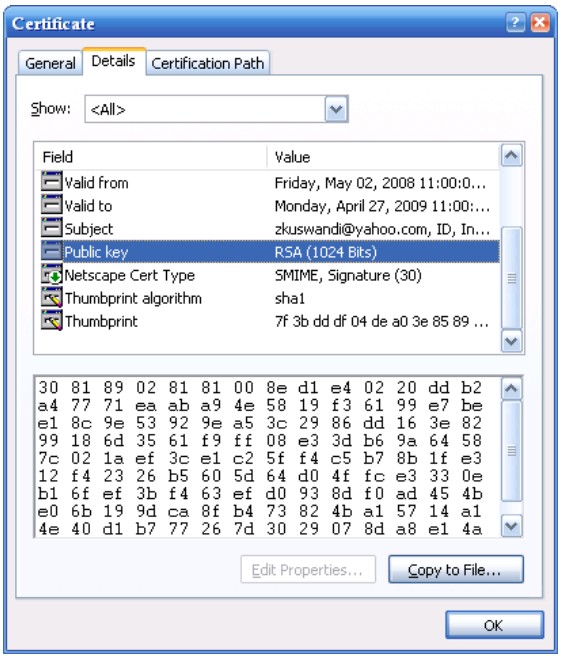
Gambar 17. Sertifikat Digital



Gambar 18. Informasi Penerbit Sertifikat

2. cekCert

Pada fungsi ini berfungsi untuk memverifikasi/ otentikasi pengguna dengan menggunakan sertifikat yang terdapat dalam aplikasi pengguna. Dimana proses yang dilakukan adalah pengguna akan mengambil sertifikat terlebih dahulu, lalu membuat *timestamp*, bilangan acak, dan tanda tangan digital. Data inilah yang akan digunakan untuk proses verifikasi sertifikat.



Gambar 19. Informasi Kunci Publik Pemegang Sertifikat

3. generateKey

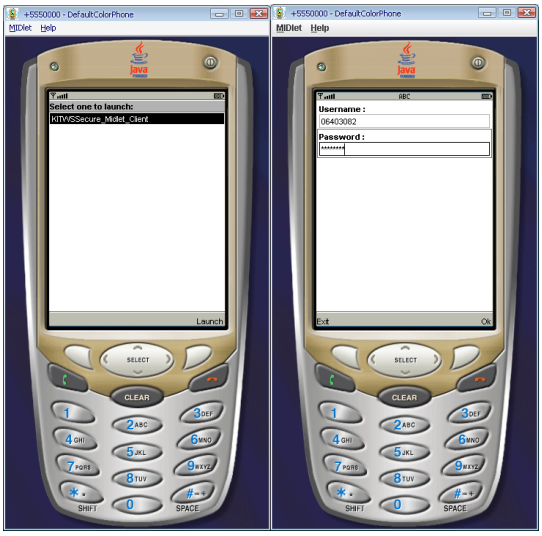
Fungsi ini berfungsi untuk menghasilkan sebuah bilangan kunci yang sama di kedua sisi (pengguna dan *web service*). Kunci ini yang akan digunakan untuk proses berikutnya, yaitu proses permintaan data atau getResData. Dimana setiap terjadi perpindahan data, akan selalu ada proses enkrip dekrip data.

4. getResData

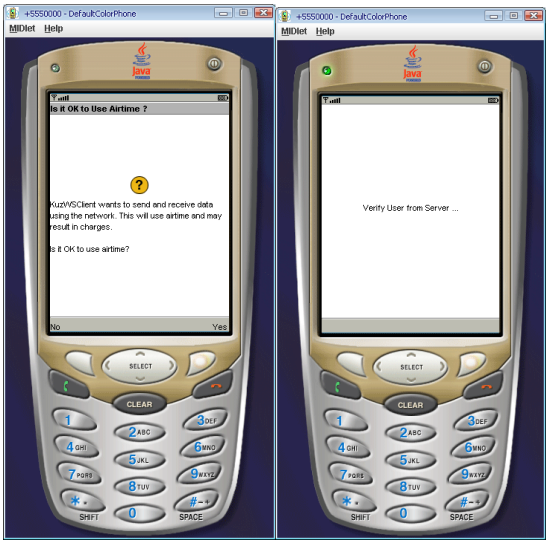
Fungsi ini berguna untuk permintaan data ke *web service*. Pada fungsi ini dibutuhkan sebuah kunci yang dihasilkan dari fungsi sebelumnya untuk proses enkrip dekrip data.

C. Uji Coba Aplikasi Pengguna

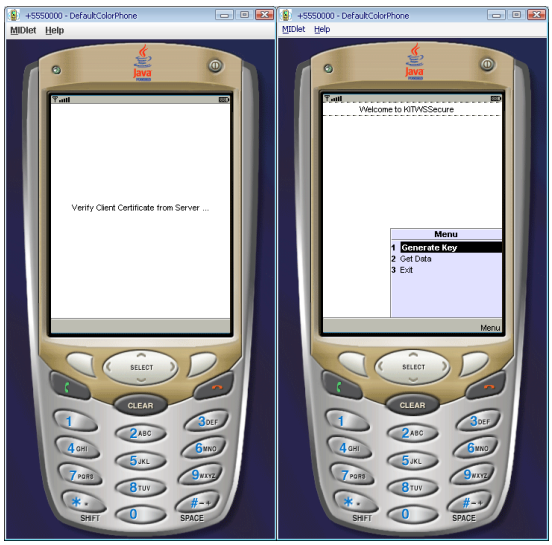
Uji coba dilakukan agar dapat diketahui apakah aplikasi yang telah dibangun sesuai dengan skenario program atau tidak. Dalam uji coba ini pengguna hanya dapat meminta satu jenis permintaan saja, sebagai contoh dari penggunaan kunci untuk mengenkrip dan mendekrip pada proses komunikasi data. Gambar 20 menggambarkan proses ketika aplikasi pertama kali dijalankan dan dilanjutkan dengan proses otentikasi pengguna pada menu login. Sistem pada *administrator* kemudian akan memeriksa apakah *username* dan *password* yang pengguna masukkan sesuai dengan data yang terdaftar pada sistem *administrator*, dengan cara menghitung dan membandingkan nilai hash *password* menggunakan algoritma SHA (Gambar 21).



Gambar 20. Tampilan awal dan menu login

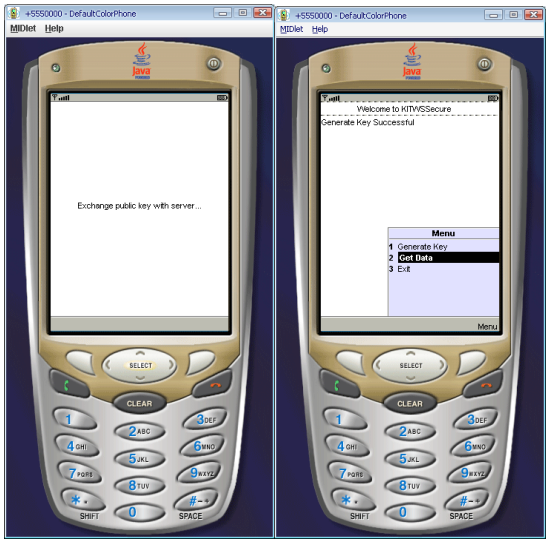


Gambar 21. Tampilan konfirmasi akses internet dan verifikasi pengguna

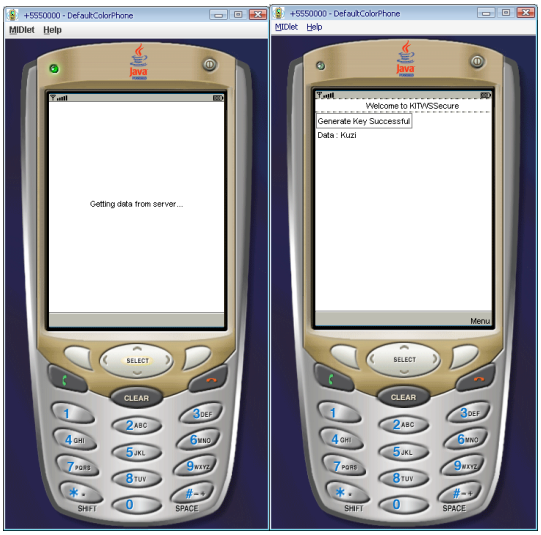


Gambar 22. Tampilan verifikasi sertifikat dan program untuk membangkitkan *session key*

Proses yang dilakukan *server* selanjutnya adalah membaca dan memverifikasi sertifikat digital pengguna. Setelah itu pengguna membuat permintaan kepada *server* untuk membentuk "*session key*" yang bersifat simetrik menggunakan algoritma Diffie-Hellman (Gambar 22).



Gambar 23. Tampilan proses pertukaran kunci dan hasil dari pertukaran kunci yang dilanjutkan pemrosesan data



Gambar 24. Tampilan proses pengambilan data dan hasil pengambilan data

Setelah *session key* terbentuk di kedua sisi (client-*server*), kemudian pengguna (client) menginisiasi proses komunikasi data (Gambar 23). Data yang dikirim dan diterima oleh kedua sisi (client-*server*) dienkripsi dan didekripsi menggunakan algoritma kunci simetrik AES, sehingga dapat dijamin proses keamanan data ketika komunikasi terjadi diantara client dan *server* (Gambar 24).

V. SIMPULAN

Berdasarkan hasil-hasil yang telah dicapai selama perancangan, pembuatan, dan pengujian perangkat lunak dalam penelitian ini, maka dapat disimpulkan sebagai berikut:

1. Sertifikat digital X.509 dapat diterapkan untuk aplikasi telepon pintar sebagai otentikasi pengguna aplikasi. Dan tahap verifikasi ini dilakukan hanya pada verifikasi kunci publik saja.
2. Sebuah sertifikat digital X.509 hanya dipegang oleh satu pengguna saja, tanpa adanya kesamaan sertifikat dengan pengguna lain. Jadi sertifikat ini bersifat unik yang hanya dimiliki oleh satu pengguna saja.
3. Pertukaran kunci antar *web service* dan pengguna dilakukan sebelum proses pemrosesan data dilakukan, dimana pertukaran kunci ini dilakukan hanya sekali saja setiap aplikasi dijalankan. Jadi setiap aplikasi dijalankan, proses pertukaran kunci akan menghasilkan kunci yang berbeda-beda untuk mengamankan pertukaran data.
4. *Web service* hanya dapat memegang satu kunci yang sama dengan pengguna, bila terdapat lebih dari satu pengguna maka *web service* akan memegang kunci yang sama dengan kunci pengguna yang terakhir melakukan proses

pertukaran kunci.

5. Proses pertukaran data dapat dilakukan berulang-ulang, karena pengguna dapat memperoleh data terbaru bila data tersebut terjadi peubahan dalam *database web service*.
6. Walaupun aplikasi *web service* dan aplikasi pengguna merupakan aplikasi yang berasal dari satu platform yaitu Java, tetapi pada aplikasi pengguna memiliki keterbatasan tipe data. Sehingga pada aplikasi pengguna harus menggunakan sebuah tools [6] agar tipe data yang tidak ada diperbolehkan untuk digunakan dalam aplikasi.

Daftar Pustaka

[1] W. Stalling, Cryptography and Network Security Principles and Practices. Fourth Edition, New Jersey: Pearson Education, Inc. 2005.

[2] W. Diffie, M. Hellman, "Multiuser Cryptographic Techniques," IEEE Transactions on Information Theory, November 1976.

[3] T. Mardiono, Membangun Solusi *Mobile* Business dengan Java. Jakarta: PT. Elex Media Komputindo,2006

[4] NIST USA, Federal Information Processing Standards (FIPS) Publication 180-2. Secure Hash Standards (SHS). 1995.

[5] NIST USA, Federal Information Processing Standards (FIPS) Publication 197. Advanced Encryption Standard (AES). 2001

[6] Eric Lafortune (diakses 2016, Februari), ProGuard, http://proguard.sourceforge.net/