هفتمین کنفرانس بین المللی انجمن رمز ایران

۲۴ و ۲۵ شهریورماه ۱۳۸۹، دانشگاه صنعتی خواجه نصیرالدین طوسی

!

انجمن رمزایران

دانشگاه صنعتی خواجه نصیرالدین طوسی

مدلی امن برای مدیریت کلید و کنترل دسترسی رمزنگاری در سیستم فایل رمزنگاری

فرشاد رحیمی اصل[۱]، رضا عزمی[۲]

تهران-دانشگاه صنعتی مالک اشتر- دانشجوی کارشناسی ارشد امنیت فناوری اطلاعات [۱]

farshad.rahimiasl@gmail.com

تهران، دانشگاه الزهرا(س)، استادیار گروه مهندسی کامپیوتر [۲]

azmi@alzahra.ac.ir

چکیده

افزایش نیاز به امنیت داده‌های کاربران در سطح سیستم عامل، باعث شده روش‌های کنترل دسترسی و سیستم فایل‌های رمزنگاری بسیاری به این منظور توسعه یابند. اما دامنه ی تهدیداتی را که روش‌های کنترل دسترسی و سیستم فایل‌های رمزنگاری موجود در بر می‌گیرند، به اندازه کافی وسیع نیستند. در این مقاله مدلی جدید برای مدیریت کلید و کنترل دسترسی رمزنگاری با ایجاد مفهوم گروه و کلاس دسترسی، برای بهینه کردن تعداد فراداده‌های رمز و هزینه‌ی حافظه و ساختار فراداده‌ها روی سیستم، ارائه شده است که این مدل حملات روی متون رمز را تا حد مطلوبی از بین می‌برد. مدیریت کلید بر اساس رمزنگاری متقارن است که باعث بالا بردن کارایی می‌شود. پارامترهایی را برای جلوگیری از حملات تکرار و جعل کاربر و انکار سرویس در محیط‌ها و ارتباطات ناامن، در نظر می‌گیریم. در این مدل بازیابی کلیدها با اعتماد کمتر به عامل بیرونی و عمل ابطال کاربران به طور بهینه انجام می‌شود و همچنین امکان رویداد نگاری امن را دارا می‌باشد.

واژه های! کلیدی

سیستم فایل رمزنگاری، واحد مدیریت کلید، گروه، کلاس دسترسی، فراداده‌ی رمز، جعبه‌ی فراداده.

۱- مقدمه !

سیستم فایل‌های رمزنگاری بسیاری به منظور ایجاد کنترل دسترسی روی داده‌های رمزنگاری در سطح سیستم عامل ایجاد شده‌اند. با توجه به اینکه در این سیستم فایل‌ها از مدل‌های ساده‌ی کنترل دسترسی استفاده می‌شود، از لحاظ امنیتی می‌توانند دچار مشکلاتی شوند که باید نوع طراحی برای این سیستم‌ها را تغییر داد به گونه‌ای که دامنه‌ی وسیعتری از تهدیدات را در برگیرند. در کنار ایجاد راهکارهای امنیتی باید توجه زیادی روی فاکتور راحتی در استفاده و کارایی برای مدل داشته باشیم. در این مقاله ما یک مدل مدیریت کلید و کنترل دسترسی برای سیستم فایل‌های رمزنگاری ارائه می‌دهیم که با توجه به در نظر گرفتن دامنه‌ی وسیعی از تهدیدات، از روش‌های بهینه در رمزنگاری استفاده می‌کند، که باعث بالا نگه داشتن کارایی مدل می‌شود. این مدل را با یک دید قابل توسعه طراحی کرده‌ایم به طوری که هیچ محدودیتی روی استفاده از نوع خاصی از الگوریتم‌های رمز، ایجاد نمی‌کند و استفاده کنندگان از این مدل می‌توانند بسته به سطح کارایی و امنیتی که می‌خواهند از الگوریتم های رمز موجود استفاده کنند.

۲-کارهای مرتبط!

سیستم فایل رمزنگاری(CFS)[1] اولین سیستم فایل رمزنگاری برای سیستم عامل یونیکس است که به عنوان یک سرور NFS در سطح کاربر پیاده‌سازی شده است. در این مدل از راهکاری برپایه‌ی کلمه‌ی عبور استفاده می‌شود که باعث کاهش کارایی و امنیت

برای اشتراک داده بین کاربران می شود. همچنین نوع طراحی، مشکل سربار کپی کردن داده و تغییر محتوا را شامل می شود.

سیستم فایل رمزنگاری شفاف(TCFS)[2] که توسعه یافته ی مدل [1] است، امکان اشتراک داده بهتری را برای کاربران فراهم می کند، ولی این مدل بر پایه ی کلمه عبور کاربران کار می کند و تمام کلید ها را در یک دایرکتوری تحت عنوان پایگاه داده ی کلید ذخیره می کند که مشکلاتی را برای امنیت سیستم فایل و داده های کاربران ایجاد می کنند.

سیستم فایل رمز شده(EFS)[3] محصول شرکت مایکروسافت است که برای سیستم عامل ویندوز و تحت سیستم فایل NTFS طراحی شده است. در این مدل برای هر فایل یک کلید در نظر می گیرند و برای فایل ها از رمزنگاری متقارن با کلید فایل و برای دسترسی کاربران به هر فایل رمزنگاری نامتقارن انجام می دهد.

eCryptfs[4] از جمله سیستم فایل های رمز نگاری ای می باشد که از روش های پیشرفته برای کنترل دسترسی استفاده می کند. در این مدل فرا داده های رمز نگاری در سرآیند فایل ذخیره شده و مدیریت کنترل دسترسی کاربران با استفاده از این فرادادە ها انجام می شود.

Transcrypt[5] مدلی است که با بهره گرفتن از روشهای پیشرفته در رمزنگاری برای ایجاد امنیت و اعمال روش های بهتر برای کنترل دسترسی نسبت به مدل های قبلی، مشکلات آنها را تاحدی حل کرده است ولی با توجه به استفاده از الگوریتم های نامتقارن برای دسترسی، کارایی این مدل تا حدی پایین می آورد.

این سیستم فایل ها دارای دامنه ی تهدید کوچک و دامنه ی اعتماد بزرگی می باشند که سیستم فایل رمز را با مشکلاتی از قبیل حملات تکرار و جعل هویت و انکار سرویس مواجه می کند. همچنین اعتماد به عاملین بیرونی از دیگر مشکلات موجود می باشد و عملیات ابطال حقوق کاربران با سربار بالایی انجام می شود. از دیگر سیستم فایل های رمز می توان به MSDOS SFS[6] و Cryptoloop[7] و BestCrypt[8] و StegFS[9] و SFS[10] و TrueCrypt و CifrarFS[11] اشاره کرد که نسبت به سیستم فایل های بالا کارایی پایین تری دارند و مشکلات آن سیستم ها را نیز شامل می شوند.[12]

۳- مدل تهدید

ابتدا یک مدل مجازی برای تهدیدات ایجاد می کنیم و سپس با توجه به فاکتورهای امنیت و راحتی در استفاده و کارایی به مقابله با تهدیدات و ایجاد مدل می پردازیم. محیط مدل، یک محیط مجازی با بیشترین میزان تهدید و کمترین میزان اعتماد است. برای طراحی مدل امنیتی هرچه میزان اعتماد کمتر باشد، مدل از امنیت بالاتری برخوردار است. تهدیدات را می توان به دو دسته تقسیم کرد: ۱- برخط. ۲- برون خط.

۳-۱ . تهدیدات برخط

تهدیدات برخط آندسته از تهدیداتی هستند که یک مهاجم برای حمله به سیستم باید به طور برخط با سیستم و کاربران و منابع اطلاعاتی در ارتباط باشد تا بتواند از این ارتباطات به نفع خود سوء استفاده کند و امنیت سیستم را مورد تهدید قرار دهد. از جمله ی این تهدیدات می توان به تهدیداتی که منجر به حملات انکار سرویس، حملات تکرار و جعل هویت می شوند، اشاره نمود.

۳-۲ . تهدیدات برون خط

نوع دیگر تهدیدات، تهدیدات برون خط هستند که یک حمله کننده نیازی به برخط بودن روی سیستم و ارتباط با کاربران ندارد. کافیست یکسری اطلاعات را از سیستم جمع آوری کند و در زمان مناسب و با قدرت محاسباتی مناسب روی آنها حمله کند و بتواند اطلاعاتی را بدست آورد. واحد مدیریت کلید را باید طوری طراحی کنیم که با نگهداری فراداده ها روی سیستم، اطلاعات خاصی را در اختیار یک مهاجم قرار ندهد.

۳-۳ . تهدیدات کاربران بدخواه

مشکل اول مسئله تبانی کاربران است. واحد مدیریت کلید باید طوری مدیریت کلید را انجام دهد که با تبانی عده ای از کاربران نشود از حقوق کاربر دیگر به نفع خود سوء استفاده کنند. در سیستم هایی که تا بحال بوده اند به این مسئله اشاره ای نشده و یک کاربر براحتی با جعل کاربر دیگر و استفاده از فراداده های رمزنگاری خود برای دسترسی به فایل ها با حق دسترسی کاربر دیگر، می تواند به نوعی یک تبانی را که در اینجا فقط خود او در آن درگیر بوده، انجام دهد.

مسئله دیگر مشکل بازیابی کلیدها است. در تمام سیستم هایی که از این مشخصه پشتیبانی کرده اند، بازیابی بر اساس اعتماد به یک عامل بیرونی انجام می شود. ولی سپردن تمام کلید ها به یک عامل بیرونی برای بازیابی، عملی مخاطره آمیز به حساب می آید که باید تا حد امکان این مسئله را حل کرد.

۴- طراحی واحد مدیریت کلید

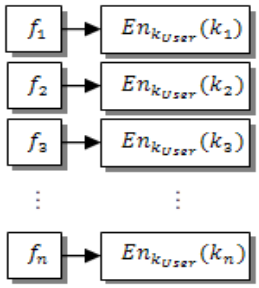
برای طراحی واحد مدیریت کلید ابتدا باید سطح ریزدانگی را مشخص کنیم. در بعضی از مدل ها مانند [1]، از ریز دانگی کمتری برای کنترل دسترسی استفاده می شود و رمز نگاری روی سطح بالاتری یعنی روی دایرکتوری انجام می شود تا اینکه روی هر فایل. این روش کارایی سیستم فایل را بدلیل اعمال کمتر عملیات رمزنگاری، بالا می برد ولی با ضعف در امنیت و کنترل دسترسی اشتراکی روی فایل مواجه می شود.

مدل های دیگرمانند [2] ،[4] و [5] برای حل این مشکل و ایجاد ریز دانگی بیشتر کنترل دسترسی را روی فایل قرار دادند، یعنی رمز نگاری روی هر دسترسی به فایل انجام می شود که این امنیت را بالا می برد ولی کارایی سیستم فایل را پایین می آورد.

باید واحد مدیریت کلید هم از ریزدانگی در سطح فایل بهره ببرد و هم کارایی خوبی داشته باشد. برای این منظور مفهوم گروه را معرفی می کنیم. در هر سیستم، مالک یک گروه، یک مجموعه فایل دارد که به هر کاربری که بخواهد اجازه دسترسی می دهد. این مجموعه فایل ها را که به یک کاربر مالک تعلق دارند، **گروه** می نامیم. این نوع تعریف از گروه ما را به سمت داشتن مزیت هایی از جمله قابلیت پیش بینی کردن دسترسی های یک کاربر می برد، که این مسئله، ما را در بالا بردن کارایی کمک می کند. مفهوم دیگری که باید روی گروه تعریف کنیم، کلاس دسترسی است. **کلاس دسترسی** روی یک گروه، یکی از زیر مجموعه های فایل های آن گروه به غیر از مجموعه ی تهی است. یک کاربر که می خواهد روی یک گروه دسترسی داشته باشد می تواند فقط در یکی از کلاس های دسترسی قرار بگیرد و با تغییر حقوق دسترسی اش بین کلاس های دسترسی جابجا شود. در بخش ۴-۱ به مزیت های استفاده از این مفاهیم برای کنترل دسترسی های رمزنگاری نسبت به مدل های موجود در این زمینه اشاره می کنیم.

۴-۱ . کاهش فراداده های رمز و هزینه ی حافظه

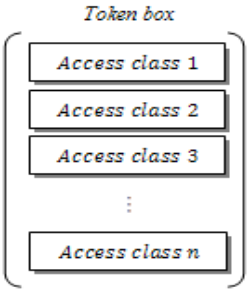
در مدل هایی با ریز دانگی در سطح فایل به ازای هر دسترسی کاربر روی فایل، یک فراداده ی رمز برای او نگه می دارند که این کار برای کنترل دسترسی های رمزنگاری مناسب نمی باشد، زیرا هم احتمال حمله روی کلید های کاربران بیشتر می شود و هم با افزایش تعداد کاربران، تعداد فراداده های رمز نیز افزایش می یابد که از لحاظ حافظه نیز ما را دچار مشکل می کند. این مسئله را در شکل ۴-۱ می توانیم ببینیم.



شکل ۴-۱. برای کاربری که به فایل های $f_1, f_2, f_3, \dots, f_n$ دسترسی دارد، باید به ازای هر فایل او یک فرا داده رمزنگاری ذخیره شود.

مساله دیگر هزینه حافظه ی فراداده های کاربران است. در این حالت برای هر مجموعه ای با N فایل و U کاربر باید به تعداد $U * N$، فرا داده ی رمز ذخیره شود که در بدترین حالت مدل ما، یعنی حالتی که دسترسی کاربران متفاوت بوده و هیچ دو کاربری مجموع دسترسی مشابه نداشته باشند، که در اینجا به تعداد $(2^N - 1)$ نوع دسترسی متفاوت وجود دارد، یعنی به تعداد زیر مجموعه های فایل ها، که اگر فرض کنیم تعداد کاربران نیز به همین تعداد باشد، ما باید به تعداد $((2^N - 1) * N)$ فراداده ی رمز ذخیره کنیم که با افزایش کاربران این تعداد نیز افزایش می یابد.[13], [14]

در طراحی مدل پیشنهادی بجای نگهداری فراداده های رمز کاربران روی هر فایل و به طور جداگانه، کلاس های دسترسی کاربران را نگه می داریم و برای همه کاربرانی که در یک کلاس دسترسی قرار می گیرند یک فراداده نگه می داریم، که این فراداده بیانگر کلاس دسترسی کاربران است. برای نگهداری کلاس های دسترسی از ساختمان داده ای به اسم جعبه فرا داده یا *token box* مطابق شکل ۴-۲ استفاده می کنیم.



شکل ۴-۲. یک جعبه ی فراداده با n کلاس دسترسی

با این روش متون رمز ما کم شده و ارتباط بین متون رمز را از بین می بریم و این مسئله احتمال حمله را برای مهاجم از بین می برد. با این روش مشکل امنیتی را که در مدل های قبل، وجود دارد را حل می کنیم. در اینجا تعداد فرا داده ی رمز در ارتباط مستقیم با تعداد کاربر یا تعداد فایل نیست، بلکه تعداد فرا داده ها ی رمز به تعداد نوع دسترسی بستگی دارد. در اینجا ما به تعداد N فایل داریم که این تعداد فایل دارای $(2^N - 1)$ زیر مجموعه غیر تهی یا کلاس دسترسی دارد. در بدترین حالت هر کاربر در یک کلاس دسترسی جداگانه قرار دارد و در این حالت ما به تعداد $(2^N - 1)$ فرا داده ی رمز نگه می داریم ولی در اینجا با افزایش کاربران دیگر تعداد فرا داده های رمز افزایش نمی یابد و همانطور که می بینیم هزینه حافظه در بدترین حالت مدل مقاله از دیگر مدل ها بهتر می باشد.

۴-۲ . ایجاد و تخصیص کلیدها

برای هر گروه یک کلید گروه KG و برای هر فایل f نیز یک کلید رمزنگاری k_f تخصیص می دهیم. در حالت اولیه ی ایجاد گروه، فقط مالک گروه وجود دارد که باید کلید او را تولید کرد و به طور امنی در اختیار او قرار دهیم.

مالک گروه دارای فایل های $f_1, f_2, f_3, \dots, f_n$ می باشد و کلید او یعنی RO از طریق رابطه ی (۱) بدست می آید:

$$RO = H(K_1, K_2, K_3, \dots, K_n) \qquad (۱)$$

اینکه ما طول کلیدها را چقدر در نظر بگیریم و از چه نوع الگوریتم چکیده سازی استفاده کنیم، به میزان امنیت و سرعتی

که می خواهیم داشته باشیم، بستگی دارد. کلید مالک گروه را تولید کرده و آن را در اولین باری که مالک تقاضا کرد به صورت رابطه ی (۲) در اختیار او قرار دهیم:

$$Owner \leftarrow En_{pu_{Owner}}(RO) \qquad (۲)$$

pu_{owner} ، کلید عمومی مالک گروه است. حالت بعدی سیستم، حالتی است که مالک می خواهد حق دسترسی روی چند فایل را به کاربر دیگری بدهد. در این حالت باید برای کاربر، کلاس دسترسی ایجاد شود و کلید آن کلاس در اختیار او قرار داده شود. کلید کاربر از طریق رابطه ی (۳) محاسبه می شود:

$$CK_{id} = H(a * k_1, a * k_2, a * k_3, \dots, a * k_n, \alpha) \qquad (۳)$$

CK_{id} نشانگر کلید کلاسی با شناسه ی id است. متغیر a می تواند 0 یا 1 باشد و این متغیر با توجه به فایل های آن کلاس 0 یا 1 می شود، طوری که دقیقا شامل کلید های متعلق به همان فایل ها باشد و k_i کلید فایل i ام می باشد و α یک عدد تصادفی است، که از این متغیر برای تغییر کلید، بعد از گرفتن حق دسترسی روی یک کلاس، برای کاربر یا کاربران، استفاده می شود. کلید کاربر را تولید کرده و با کلید عمومی او رمز کرده تا هر وقت کاربر برای اولین بار آن را درخواست کرد، کلید را به او بدهیم که این مرحله را از طریق رابطه ی (۴) انجام می دهیم:

$$User \leftarrow En_{pu_{User}}(CK_{id}) \qquad (۴)$$

مسائلی که در سیستم فایل های موجود مشاهده می شود، مربوط به بحث حمله و پنهان نگه داشتن کلیدها حتی از کاربران مجاز می باشد. برای جلوگیری از حملات تکرار و ایجاد تازگی و پنهان نگه داشتن کلید ها از سطح کاربر دو مقدار تصادفی را در نظر می گیریم. با این اطلاعات فیلد مربوط به سطح کاربر در جعبه ی فراداده را می سازیم که این فیلد در رابطه ی (۵) نشان داده شده است. فرض می کنیم گروه شامل n فایل است:

$$En_{CK_{id}}(En_R(a * k_1, a * k_2, a * k_3, \dots, a * k_n), r) \qquad (۵)$$

a می تواند 0 یا 1 باشد و پارامتر CK_{id} کلید کلاس با شناسه ی id است و پارامتر R عدد تصادفی برای پنهان سازی مقادیر کلیدهای فایل ها است و r عدد تصادفی برای جلوگیری از حمله تکرار می باشد. قسمت فراداده ی کلیدها یکبار به ازای ایجاد کلاس دسترسی تولید می شود و فقط مقدار تصادفی r تغییر می کند. نکته مهم که هم در کارایی و هم در امنیت دخیل است مربوط به این است که چه موقع از کاربر بخواهیم که اصالت درخواست خود را نشان دهد که این ملزم به فرستادن فراداده به سطح کاربر و رمزگشایی کاربر می باشد و بعد از هر مرحله باید مقدار تصادفی r نیز عوض شود. در اینجا ما این مسئله را به عهده ی استفاده کنندگان این واحد مدیریت کلید می گذاریم که با توجه به میزان امنیت و کارایی که می خواهند، می توانند تعداد دفعات و زمانهای این احراز اصالت درخواست ها را مشخص کنند.

سه پارامتر r، R و α را باید در سطح سیستم به صورت رمز شده نگهداری کنیم و مرحله تولید کلید باید طوری باشد که سیستم نخواهد کلیدها را نزد خود نگهداری کند تا از لحاظ امنیتی دچار مسئله ای نشود. فیلد مربوط به واحد مدیریت کلید از طریق رابطه ی (۶) تشکیل می شود:

$$En_{C_{id}}(R, r, \alpha) \qquad (۶)$$

البته برای فراداده ی مالک گروه نیازی به نگهداری پارامتر α نیست، چون کلید مالک تغییر نمی یابد و لزومی ندارد در پروسه تغییر کلید و تجدید آن شرکت کند و فیلد دوم آن به صورت رابطه ی (۷) است:

$$En_{C_{id}}(R, r) \qquad (۷)$$

پارامتر C_{id} که برای رمز نگاری در رابطه ی (۶) و (۷) استفاده می شود، از طریق رابطه ی (۸) ساخته می شود:

$$C_{id} = H(H(U_1, U_2, U_3, \dots, U_l), KG) \qquad (۸)$$

id شناسه مربوط به کلاس دسترسی ای می باشد که فیلد (۸) متعلق به آن است. H تابع چکیده ساز می باشد و l تعداد کاربران کلاس دسترسی می باشد. این روش ساخت کلید، ما را قادر می سازد که در موقع نیاز با یک الگوریتم سریع کلید ها را تولید کنیم ولی نیازی به نگهداری کلید ها در سیستم نمی باشد. فیلد سوم که در تولید C_{id} دخالت دارد، چکیده کاربران عضو آن کلاس دسترسی می باشند. که با رابطه ی (۹) در ابتدای تخصیص دسترسی به l کاربر کلاس، تولید و نگهداری می شود:

$$H(U_1, U_2, U_3, \dots, U_l) \qquad (۹)$$

در این مدل برای کنترل دسترسی رمزنگاری از یک ماتریس دسترسی و یک جعبه ی فراداده استفاده می کنیم. شکل ۴-۳ یک ماتریس کنترل دسترسی روی فایل ها برای کاربران است. ستون های آن نشانگر فایل و سطرهای آن نشانگر کاربران است. ستون آخر بیانگر یک اشاره گر برای کاربر به کلاس دسترسی مربوط به خود، در جعبه ی فراداده است. هر عنصر این ماتریس حقوق (r)کاربر مربوط $(User)$ را روی فایل مربوطه(f) نشان می دهد که به صورت $(r_{User})_f$ بیان می شود. در کنار این ماتریس یک مولفه ی دیگر یا همان جعبه ی فراداده قرار دارد که متون رمز کلاس های دسترسی را در خود نگه می دارد که ساختار آن به صورت شکل ۴-۴ است.

شکل ۴-۳. ماتریس کنترل دسترسی

$$\begin{array}{c|cccccc} & f_1 & f_2 & f_3 & \cdots & f_n & Class\ reference \\ Owner & (r_0)_{f_1} & (r_0)_{f_2} & (r_0)_{f_3} & \cdots & (r_0)_{f_n} & Owner\ class \\ User_1 & (r_{U_1})_{f_1} & (r_{U_1})_{f_2} & (r_{U_1})_{f_3} & \cdots & (r_{U_1})_{f_n} & User_1\ class \\ User_2 & (r_{U_2})_{f_1} & (r_{U_2})_{f_2} & (r_{U_2})_{f_3} & \cdots & (r_{U_2})_{f_n} & User_2\ class \\ User_3 & (r_{U_3})_{f_1} & (r_{U_3})_{f_2} & (r_{U_3})_{f_3} & \cdots & (r_{U_3})_{f_n} & User_3\ class \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ User_l & (r_{U_l})_{f_1} & (r_{U_l})_{f_2} & (r_{U_l})_{f_3} & \cdots & (r_{U_l})_{f_n} & User_l\ class \end{array}$$

Class address	Cryptographic tokens
Owner class	$En_{RO}\left(En_{R_0}(k_1,k_2,k_3,\dots,k_n),r_0\right),En_{C_0}(r_0,R_0),H(Owner)$
First class	$En_{CK_1}\left(En_{R_1}(a*k_1,a*k_2,a*k_3,\dots,a*k_n),r_1\right),En_{C_1}(r_1,R_1,\alpha_1),H(First\ class's\ users)$
Second class	$En_{CK_2}\left(En_{R_2}(a*k_1,a*k_2,a*k_3,\dots,a*k_n),r_2\right),En_{C_2}(r_2,R_2,\alpha_2),H(Second\ class's\ users)$
Third class	$En_{CK_3}\left(En_{R_3}(a*k_1,a*k_2,a*k_3,\dots,a*k_n),r_3\right),En_{C_3}(r_3,R_3,\alpha_3),H(Third\ class's\ users)$
⋮	⋮
i'th class	$En_{CK_i}\left(En_{R_i}(a*k_1,a*k_2,a*k_3,\dots,a*k_n),r_i\right),En_{C_i}(r_i,R_i,\alpha_i),H(i'th\ class's\ users)$

شکل ۴-۴. جعبه ی فراداده

۳-۴ . بهبود در عمل بازیابی

در سیستم فایل های رمزنگاری موجود، آنهایی که از عمل بازیابی پشتیبانی می کنند، مدیریت کلید آنها طوری طراحی شده است که در هنگام دادن حق دسترسی به کاربران، یک حق دسترسی روی همان داده ها را نیز برای عامل بازیاب ایجاد می کنند که بتوانند در مواقع لازم، کلید های کاربران و فایل ها را بازیابی کنند [12]. ولی این روشها چون بر یک عامل بیرونی تمرکز دارند، نمی توانند امنیتی را که می خواهیم برای ما ایجاد کنند.

در مدل پیشنهادی در این مقاله، همانطور که در قسمت ۴-۲ دیدیم، کلید های کاربران طوری تولید می شوند که تمام کلید های کاربران را مالک گروه با کمک واحد مدیریت کلید می تواند بازیابی کند. بازیابی کلید مالک گروه کمی فرق دارد، چون روش تولید کلید مالک گروه متفاوت بوده و باید برای بازیابی آن تمام کلید ها را در اختیار داشت تا بتوان از فرمول (۱)، استفاده کرد. اگر در گروه به تعداد لازم کلاس دسترسی داشته باشیم، به طوری که اجتماع دسترسی های آنها، دسترسی روی کل گروه را بسازد، آنگاه می توان کلید مالک گروه را با کمک کاربران آن کلاس ها و واحد مدیریت کلید، بازیابی کرد. اما تا زمانی که این شرط برقرار نشود باید از یک عامل بیرونی برای این کار کمک بگیریم. روش کار اینچنین است که در ابتدای ایجاد گروه و ایجاد کلید مالک، یک عامل معتمد بیرونی را برای بازیابی در یک کلاس دسترسی قرار می دهیم، به طوری که روی تمام فایل ها با کمترین مجوز، دسترسی دارد. با اضافه شدن کاربری به گروه، فایل هایی را که کاربر در اختیار دارد، از اختیار عامل بازیاب بیرونی در می آوریم و کلاس دسترسی او را تغییر می دهیم و این کار را آنقدر می توانیم ادامه دهیم که عامل بازیاب بیرونی کامل از گروه خارج شود و خود کاربران گروه و واحد مدیریت کلید، بازیابی کلید مالک را انجام دهند.

۴-۴ . بهینه سازی در عمل ابطال حقوق

در سیستم فایل های موجود با توجه به اینکه مدیریت کلید برمبنای تک کلید کاربر است، با گرفتن حق دسترسی از کاربر طوری عمل می کنند که کاربر از فراداده های رمز قبلی خود نتواند استفاده ای کند. اما این کار ملزم به ایجاد کلید های جدید برای فایل ها و رمزنگاری مجدد روی فایل ها و بروز کردن فراداده های کاربران دیگر است تا بتوانند فرا داده ی کاربر باطل شده را کاملاً بیهوده سازند. در صورتی که در بسیاری از سیستم ها عمل ابطال را بدون توجه به این مسئله ی امنیتی انجام می دهند.

در مدل مقاله، اگر بخواهیم کاربری را از کلاس دسترسی ای خارج کنیم، فقط لازم است کلید کلاس دسترسی را تغییر دهیم. از پارامتر α در رابطه (۳) برای تغییر کلید و باطل کردن آن و ساخت کلید جدید استفاده می کنیم. کاربر خارج شده، چون از کلید فایل ها اطلاعی ندارد، هیچگونه سوء استفاده ای از دسترسی قبلی خود نمی تواند داشته باشد. کاربران دیگر هر وقت بخواهند به کلاس، دسترسی داشته باشند، کافی است یکبار کلید جدید کلاس دسترسی را دریافت کنند. همانطور که می بینیم در این مدل بجای انجام تعداد زیادی عملیات رمز روی داده ها در سطح سیستم، فقط کافی است کلیدِ یک کلاس دسترسی را بروز کنیم.

به نوعی از اعمال تغییرات سنگین و زیاد در سطح سیستم جلوگیری به عمل می آید.

۴-۵ . جامعیت داده

برای ایجاد جامعیت فایل ها، کاربرانی که مجاز به تغییر روی فایل هستند، باید پس از ایجاد تغییرات در فایل، روی چکیده ی فایل امضا کنند. البته این امضا را برای بالا بردن کارایی، خود واحد مدیریت کلید می تواند تولید کند ولی به منظور مسئله ی تفکیک وظایف و مسائل امنیتی دیگر، بهتر است خود کاربران این امضا را تولید کنند. با هر تغییر در ماتریس دسترسی و جعبه ی فرا داده، مالک گروه باید روی چکیده ی آنها، امضا کند. در اینجا چگونگی امضا و الگوریتم ها و مراحل تولید امضا بحث نمی شود ولی این مراحل می توانند بدون ایجاد سربار برای سطح کاربر و در زمان کم بار سیستم انجام شوند.

۴-۶ . تفکیک وظایف و رویدادنگاری

در سیستم فایل های موجود چون روی هر درخواست، عملیات احراز اصالت کاربر انجام می شود، تفکیک روی حقوق پایین نیز انجام می شود و این همیشه ثابت است و تغییر نمی کند. در این سیستم فایل ها، چون مستعد حملات تکرار و جعل کاربر هستند، نمی توان رویداد نگاری امنی انجام داد.

مدل پیشنهادی ما چون بر اساس گروه و کلاس دسترسی است، این انعطاف را دارد که بتوان برای آن سطح تفکیک تعیین کرد و رویداد نگاری امنی انجام داد. مثلاً برای یک استفاده کننده از این مدل، می تواند برای کاربران یک کلاس دسترسی، سطح تفکیک را روی حقوق بالاتر تعیین کند. همانطور که در بخش ۵ اشاره خواهد شد، یک پارامتر تعیین می کنیم و بر اساس آن احراز اصالت کاربر و درخواست او را انجام می دهیم. اما برای احراز اصالت کافی است فراداده ای را که به او برای رمز گشایی در بار اول می دهیم، با امضای خود، آن را پاسخ دهد،که با رابطه ی (۱۰) می تواند اینکار را انجام دهد.

$$S_{User}(En_R(Class's\ files\ key), r) \qquad (۱۰)$$

در ابتدا فراداده ی رمز به سطح کاربر فرستاده می شود و کاربر باید آنرا رمز گشایی کند و با امضای خود روی آن، به سطح واحد مدیریت کلید بفرستد. S_{User} امضای کاربر روی فراداده ی رمزگشایی شده است. با این روش هم یک رویدادنگاری امن ایجاد می کنیم و هم جلوی حملاتی از نوع انکار سرویس و جعل کاربر را در این مرحله از ارتباط می گیریم.

۴-۷ . کارایی مراحل تولید کلید

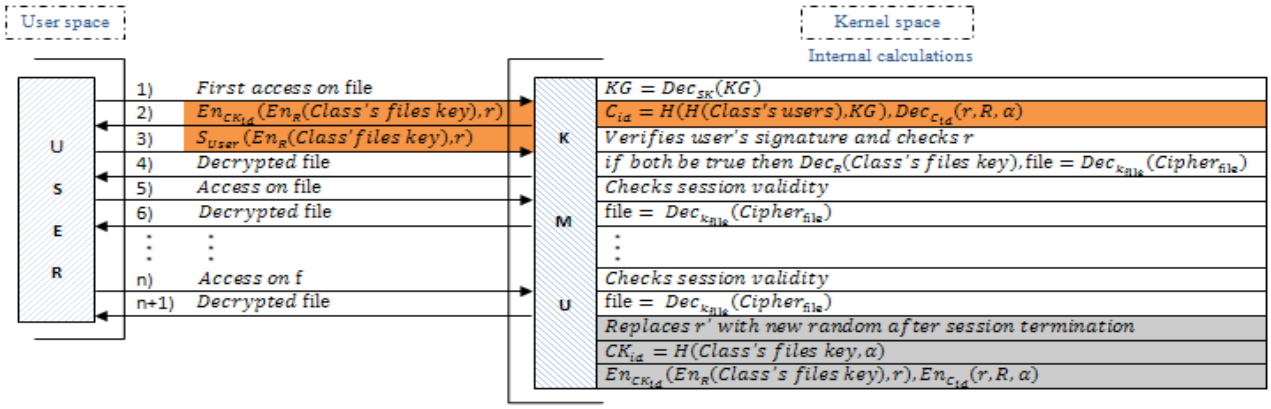
همانطور که در بخش ۴-۲ بیان شد، به ازای هر کلاس دسترسی یکسری کلید و فراداده تولید و نگهداری می شود. اما با توجه به بخش ۴-۱ تعداد این فراداده ها در بدترین حالت نیز از مدل های دیگر کمتر می باشد و کلید های سطح کاربر یکبار به ازای ایجاد کلاس دسترسی یا تجدید کلید در مرحله ی ابطال حقوق، تولید می شوند و می توان از سربار آنها صرف نظر کرد. کلید های سطح سیستم به ازای درخواست های کاربر روی کلاس دسترسی تولید می شوند و این عملیات با توجه به شکل ۵-۱ به موازات عملیات سطح کاربر انجام می شود و زمان انجام این عملیات که می تواند با الگوریتم های سریع متقارن باشد، در زمان کل همپوشانی داشته باشد و سرباری را برای سیستم ایجاد نکند.

کلید های فایل ها نیز یکبار به ازای تولید گروه ایجاد می شوند و تا پایان عمر گروه ثابت می مانند و سرباری را متوجه عملیات کاربران نمی کند. البته برای بالا بردن امنیت بهتر است در فواصل زمانی فراداده ها و کلید های گروه بروز و تجدید شوند.

۵- ارتباط با واحد مدیریت کلید

در سیستم فایل های رمزنگاری پیشرفته مانند [4] و [5]، برای ارتباط کاربر با واحد مدیریت کلید و مدیریت درخواست های او، برای هر درخواست روی هر فایل، کاربر را به یک عمل رمزنگاری نامتقارن مجبور می کنند که این مسئله از لحاظ کارایی مناسب نمی باشد. به نوعی به دلیل ازدیاد فراداده ها و نیاز به رمزگشایی کاربر برای احراز اصالت درخواست، سطح کاربر نیاز به رمزنگاری های متعدد با هزینه ی بالا می باشد. همچنین به دلیل مستعد بودن حملات تکرار و جعل کاربر در مدل های موجود نمی توان رویداد نگاری امنی انجام داد و در صورت ناامن بودن ارتباط سطح کاربر و هسته، مهاجمین می توانند اطلاعات زیادی را بدست بیاورند.

مدل مقاله چون بر اساس گروه و کلاس دسترسی کار می کند، می توان سربار انجام عمل رمزنگاری را روی هر درخواست فایل حذف کرد و این کار را روی درخواست های یک گروه انجام داد. ولی مدل این انعطاف را دارد که ما می توانیم اینکار را روی هر تغییر در شناسه ی پروسه ی کاربر یا روی تعداد فایل های درخواستی انجام دهیم و بسته به اینکه ما چه سطح از کارایی و امنیت را می خواهیم، می توانیم این پارامتر را تعیین کنیم. در شکل ۵-۱، روند ارسال درخواست ها و فراداده ها نشان داده شده است. ابتدا با توجه به درخواست کاربر و حقوق تعریف شده برای او در ماتریس کنترل دسترسی، اشاره گر مربوط به کلاس دسترسی او بازیابی شده و سپس فراداده ی سطح کاربرِکلاس دسترسی را برای احراز اصالت درخواست به سطح کاربر می فرستد و کاربر فقط یک بار در مرحله ی ۳، از امضا استفاده می کند.



شکل ۵-۱. ارتباط با واحد مدیریت کلید

از مرحله ی ۵ به بعد تا شروع نشست بعدی، دیگر عملیات رمزنگاری در سطح کاربر لازم نیست و این مسئله کارایی را تا حد خوبی بالا می آورد. کلید SK کلید مخفی واحد مدیریت کلید است و می توان برای نگهداری امن آن از یک ماجول پایگاه معتمد (Trusted Platform Module) استفاده کرد و یا به صورت احراز هویت با یک پسورد که کلید SK از آن بدست می آید و می توان چکیده ی آن را در بلاک برتر سیستم فایل ذخیره نمود. مراحلی که با رنگ نارنجی نشان داده شده است، مربوط به رمزگشایی فراداده ی سطح کاربر و تولید کلید های مورد نیاز سطح هسته می باشد که این مراحل به طور موازی انجام می شوند. مراحلی که با خاکستری نشان داده شده، در پایان نشست برای بروز کردن فراداده ها با مقدار تصادفی جدید انجام می شوند و هیچ سربراری را متوجه سمت کاربر نمی کند و این مراحل می توانند در زمان مناسب و کم بار سیستم انجام شوند که در بالا بردن کارایی نقش مهمی دارند. همانطور که در شکل ملاحظه می کنید برای پاسخ دادن به درخواستهای کاربر طی یک نشست از رمز متقارن بهره گرفته ایم که کارایی مدل را نسبت به اکثر مدل های پیشرفته ی موجود بالا می برد.

۶- ارزیابی مدل

به منظور طراحی مدل مقاله، سه فاکتور کلی امنیت و راحتی در استفاده و کارایی را در نظر گرفته ایم و بر اساس این سه فاکتور، معیار های دیگری را استخراج کرده ایم که در زمینه ی امنیت سیستم فایل لازم می باشند. ارزیابی مدل را بر اساس این فاکتورها در نظر گرفته ایم و مدل مقاله را با دیگر مدل ها مقایسه نموده ایم که می توان این مقایسه را در جدول ۶-۱ ملاحظه نمود. همانطور که در بخش ۴-۳ اشاره شد، مدل مقاله عملیات بازیابی را با اعتماد کمتر به عامل بیرونی انجام می دهد. به نوعی عامل بیرونی را از دامنه ی اعتماد خارج کرده است و عملیات بازیابی با کارایی بالاتری نسبت به مدل های قبلی انجام می شود و این مسئله را می توان از تعداد عملیات رمز کمتر در مدل مقاله متوجه شد.

در این مدل امکان هرگونه تبانی از کاربران گرفته شده است و با توجه به وابستگی تولید کلید به کلید های فایل ها و یکسری اطلاعات مخفیِ دیگر و عدم اطلاع کاربران از محتویات کلید فایل ها و دیگر اطلاعات مخفی، کاربران نمی توانند با تبانی همدیگر به تولید کلید بپردازند و از منابعی که در اختیار دارند نمی توانند اطلاعاتی را استخراج کنند که در این مدل تمام این نیازهای امنیتی با توجه به روند تولید کلید برآورده می شوند. این موضوع را می توان در روابط بخش ۴-۲ ملاحظه نمود. با توجه به مباحث گفته شده و اطلاعات جدول ۶-۱ ملاحظه می کنید که در طراحی مدل هم به اصول امنیتی توجه زیادی شده و هم به مقابله با تهدیداتی که می تواند متوجه مدل باشد پرداخته شده است.

برای فاکتور راحتی در استفاده، باید معیارهایی را در نظر بگیریم و عملکرد مدل را با آن معیارها مقایسه کنیم. این معیارها شامل شفاف بودن عملیات رمز از دید کاربر و درگیر نبودن کاربر در انجام عملیات و ذخیره ی داده های زیادی از اطلاعات رمز است. در این مدل با توجه به اینکه تمام عملیات رمز روی داده ها در سطح واحد مدیریت کلید انجام می شود و کاربر با آن درگیر نمی باشد، از شفافیت خوبی برای سطح کاربر برخوردار می باشد.

جدول ۶-۱. مقایسه مدل تحقیق

خاصیت	MSDOS SFS	EFS	CFS	TCFS	eCryptfs	TransCrypt	SFS	مدل مقاله
محرمانگی	دارد	دارد	دارد	دارد	دارد	دارد	دارد	دارد
جامعیت	ندارد	ندارد	ندارد	دارد	دارد	دارد	دارد	دارد
دسترس پذیری	ندارد	ندارد	ندارد	ندارد	ندارد	ندارد	ندارد	دارد
مدیریت کلید	هر بخش و هر کاربر	هر فایل و هر کاربر	هر دایرکتوری و هر کاربر	هر فایل و هر کاربر	هر فایل و هر کاربر	هر فایل و هر کاربر	هر فایل و هر کاربر	هر فایل و هر کاربر
اشتراک امن	ندارد	دارد	ندارد	دارد	دارد	دارد	دارد	دارد
بازیابی امن	ندارد	ندارد	ندارد	ندارد	ندارد	ندارد	ندارد	دارد
عملیات لغو کارا	ندارد	ندارد	ندارد	ندارد	ندارد	ندارد	ندارد	دارد
رویداد نگاری	دارد	دارد	دارد	دارد	دارد	دارد	دارد	دارد
کاربر برتر	معتمد	معتمد	معتمد	معتمد	معتمد	معتمد	معتمد	نامعتمد

برای هر گروه یک کلید در نظر می گیریم و کاربر به تعداد گروه هایی که دارد، کلید کلاس دسترسی در اختیار دارد که این تعداد کلید برای سطح کاربر منطقی می باشد و همچنین یکسری از مزیت های امنیتی را که کاربر می خواهد برای او فراهم می کند و کاربر می تواند با توجه به اهمیت گروه برای او به ذخیره سازی کلید های خود بپردازد و از دست رفتن یک کلید، فقط داده های مربوط به آن کلید را متوجه از دست رفتن می کند و دیگر گروه های کاربر از این از دست رفتن مصون می مانند. با توجه به این معیارها، این مدل برای کاربران می تواند از شفافیت خوبی برخوردار باشد و از لحاظ راحتی در استفاده در سطح خوبی قرار بگیرد. فاکتور دیگری که در تمام مراحل طراحی مدل در نظرگرفته شده، فاکتور کارایی می باشد که تا حد امکان از انجام عملیات رمز نامتقارن و سنگین در مدل خودداری شده است و تعداد عملیات رمز و کنترل دسترسی را به حداقل رسانده ایم و می توان این نتیجه را در مرحله ی ابطال حقوق کاربران ملاحظه کرد که نسبت به مدل های قبل از کارایی بالاتری برخوردار می باشد.

۷- نتیجه گیری

سیستم فایل های موجود از روش هایی استفاده می کنند که مشکلات امنیتی و کارایی مختلفی دارند. در اینجا ما مدلی ارائه دادیم که در کنار اعمال مسائل امنیتی توجه زیادی روی فاکتور راحتی در استفاده و کارایی کرده ایم. با این مدل جلوی حملاتی از نوع تکرار و جعل کاربر و انکار سرویس را در مراحلی از ارتباطات گرفتیم و مدل را طوری طراحی کرده ایم که این انعطاف را دارد که با توجه به میزان کارایی که می خواهیم، می توانیم از مولفه های امنیتی استفاده کنیم. همچنین خصوصیاتی از جمله بازیابی امن داده و رویداد نگاری امن و بهینه سازی عمل ابطال کاربر و جلوگیری از تبانی را شامل می شود. از تحقیقات بعدی که می توان در این حوزه انجام داد، می توان به بحث تولید کلید اشاره کرد. اگر بتوان روند ساخت کلید ها را با الگوریتم های بهینه انجام داد می توان کارایی را بالاتر ببریم. از دیگر کارهای آینده می توان به بحث بازیابی امن اشاره کرد. اگر بتوانیم روند ساخت کلید ها را به گونه ای انجام دهیم که دیگر نیازی به کمک گرفتن از عامل بیرونی برای عمل بازیابی در هیچ مرحله ای نداشته باشیم، می توانیم امنیت مدل را بالاتر ببریم.

مراجع

[1] M. Blaze. “A cryptographic file system for Unix,” In Proceedings of the first ACM Conference on Computer and Communications Security, (1993).

[2] G. Cattaneo, L. Catuogno, A. Del Sorbo, and P. Persiano. “The Design and Implementation of a Transparent Cryptographic Filesystem for UNIX,” In Proceedings of the Annual USENIX Technical Conference, FREENIX Track, pages 245–252, (June 2001).

[3] Microsoft Corporation. “How Encrypting File System Works,”http://technet.microsoft.com/en-us/library/cc781588%28WS.10%29.aspx, (November 11, 2009)

[4] Mike Halcrow. “ecryptfs: a stacked cryptographic filesystem,” Linux J., 2007(156):2, (2007).

[5] Satyam Sharma, Rajat Moona, and Dheeraj Sanghi. “Transcrypt: A secure and transparent encrypting file system for enterprises,” In Proceedings of the 8th International Symposium on Systems and Information Security, SSI 2006, Sao Paulo, Brazil, (November 2006).

[6] P. C. Gutmann. Secure filesystem (SFS) for DOS/Windows.http://www.cs.auckland.ac.nz/~pgut001/sfs/ ,(1994).

[7] Ralf Hlzer. Cryptoloop. http://www.tldp.org/HOWTO/Cryptoloop-HOWTO/, (2004).

[8] Mick Bauer. Paranoid penguin: “Bestcrypt: cross-platform filesystem encryption,” Linux J., 2002(98):9, (2002).

[9] A. D. McDonald and M. G. Kuhn. “StegFS: A Steganographic File System for Linux,” In Information Hiding, pages 462–477, (1999).

[10] Rajesh Kumar Pal. “Design and Implementation of Secure FileSystem,” M.TechThesis, Indian Institute of Technology Kharagpu, (May 2008).

[11] Kuala Lumpur, Malaysia. “CifrarFS - Encrypted File System using FUSE”, International Journal of Computer Science and Security (IJCSS), pages 295 – 302,(July/August 2009).

[12] C.P.Wright, J.Dave, and E.Zadok. “Cryptographic File Systems Performance: What You Don’t Know Can Hurt You,” In Proceedings of the 2003 IEEE Security in Storage Workshop (SISW2003), pages47 61, Washington, DC, (October2003).

[13] Abhijit Bagri. “Key Management for TransCrypt,” Master's thesis, Indian Institute of Technology Kanpur, India, (May2007).

[14] S. Thyregod. “Key management in cryptographic access control,” Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, (2006).